A Planned Approach to High Collision Risk Area

by

John Zhang Li

B.S. & M.S., Embry-Riddle Aeronautical University, 2013

Submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

and the

WOODS HOLE OCEANOGRAPHIC INSTITUTION

September 2020

Author .....

Joint Program in Applied Ocean Science & Engineering
Massachusetts Institute of Technology
& Woods Hole Oceanographic Institution
August 07, 2020

Certified by .....

Michael R. Benjamin
Principal Research Scientist, Department of Mechanical Engineering
Thesis Supervisor

Certified by .....

John J. Leonard
Samuel C. Collins Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by .....

Nicolas Hadjiconstantinou
Chairman, Committee for Graduate Students
Massachusetts Institute of Technology

Accepted by .....

David Ralston
Chairman, Joint Committee for Applied Ocean Science & Engineering
Woods Hole Oceanographic Institution

A Planned Approach to High Collision Risk Area

by

John Zhang Li

Submitted to the Joint Program in Applied Ocean Science & Engineering
Massachusetts Institute of Technology
& Woods Hole Oceanographic Institution
on August 07, 2020, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

This thesis examines the transition of a vessel from the open ocean, where collisions are rare, to a high risk and heavy traffic area such as a Traffic Separation Scheme (TSS). Previous autonomy approaches generally view path planning and collision avoidance as two separate functions, i.e. a vessel will follow the planned path until conditions are met for collision avoidance algorithms to take over. Here an intermediate phase is proposed with the goal of adjusting the time of arrival to a high vessel density area so that the risk of collision is reduced. A general algorithm that calculates maximum future traffic density for all choices in the speed domain is proposed and implemented as a MOOS-IvP behavior. This behavior gives the vessel awareness of future collision risks and aids the collision avoidance process. This new approach improves the safety of the vessel by reducing the number of risky encounters that will likely require the vessel to maneuver for safety.

Thesis Supervisor: Michael R. Benjamin
Title: Principal Research Scientist, Department of Mechanical Engineering

Thesis Supervisor: John J. Leonard
Title: Samuel C. Collins Professor of Mechanical and Ocean Engineering

Acknowledgments

As a first generation immigrant, I am immensely grateful to this country for the refuge and freedom it has offered my family. I want to thank the United States Navy for the opportunity to serve, and the support given to pursue research through the MIT-WHOI Joint Program. Working and interacting with eager students and leading experts at two world-class institutions has been a truly exhilarating and humbling experience.

This work wouldn't have been possible without the enduring support of my research advisors Dr. Michael Benjamin and Dr. John Leonard. Thank you John for making Marine Robotics Group a welcoming place since day one, and giving me the academic freedom to pursue my own topics of interests. Thank you Mike for the incredible patience in teaching me the skills needed. When I had doubts, your friendship and encouragements made a huge difference.

Thank you to my fellow Navy students at the Joint Program for your friendship. You are consummate professionals and always lead by example. To my labmates at the Marine Robotics Group, thank you for making me part of the community. Special thanks to Kevin Doherty, Alan Papalia, and Antonio Teran for always taking the time to help.

Thank you to all of the MIT/WHOI staff, and specifically Dr. Andone Lavery, Dr. Henrik Schmidt, and Kris Kipp for their support and guidance upon arriving to the Joint Program. They helped and supported a smooth transition to a challenging academic setting from a life at sea.

A final thank you to my parents for their love and support from afar, and to the love of my life, Iqra, for her daily love and encouragement right beside me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent twin U.S. Navy ship collisions have shocked the maritime community and provoked a deep reflection in the service regarding the training and practice of seamanship in today's maritime environment [37]. Coastal regions, sea lanes, and harbors are becoming increasingly congested as the need for marine transportation skyrockets to facilitate the rise in global commerce. Among severe congestion, even experienced operators can make mistakes being overwhelmed with information from radar, automatic identification system (AIS), electronic charts and other systems. In contrast, the high seas, ocean beyond the 200 nautical miles limit of national jurisdictions, are marked by such sparse traffic that can lull the most alert mariner into a false sense of safety. The navigational environment, therefore, is for the most part dull but can be interspersed with periods of extreme peril. Since robots have historically been designed to solve problems that are dull and/or dangerous, Unmanned Surface Vehicles (USVs) could become a contributor to safer navigation. But can USVs overcome the challenges presented by such diverse and complex maritime environment?

To start, a USV must be able to get from point A to point B, while avoiding obstacles along the way. To accomplish this task, years of research and development have produced both efficient global path planners and fast reactive local path planners. An efficient global path planner considers static layouts such as land, buoys, and navigational hazards to form a suitable path often represented by waypoints constrained by mission requirements [32]. A local path planner, synonymous with collision avoid-

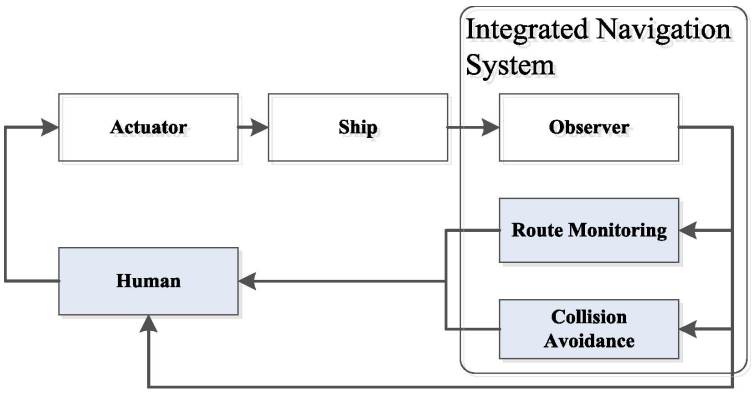
ance algorithms in the maritime context, maneuvers the USV to safety in reaction to moving obstacles such as another vessel, or a newly detected hazard, often having to temporarily disregard the requirements of the mission [49]. While both essential, the studies of global path planning and collision avoidance have diverged. A renewed emphasis on the bridge between the two could improve the safety of not only USVs but also manned vessels that might interact with them.

1.1 Safety of Navigation

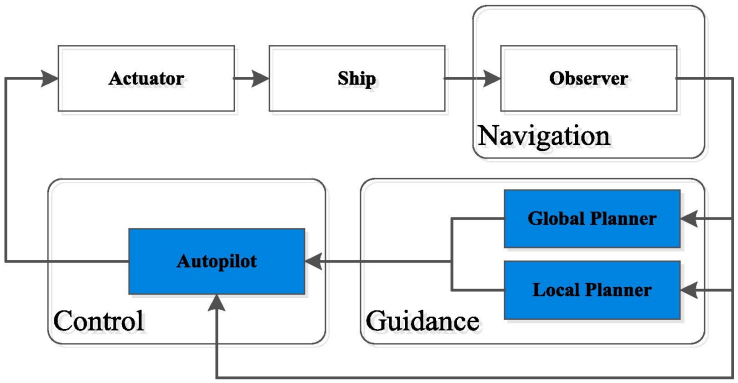
An examination of how manned and unmanned vessels approach safety of navigation reveals a natural dichotomy between larger scale path planning and reactive collision avoidance [22], as shown in Figure 1-1.

For manned ships, the normal mode of operation is route monitoring and following. Typically a pre-planned and pre-approved navigation plan is entered into the ship's bridge system such as an Integrated Navigation System (INS). This navigation plan at a minimum takes consideration of the mission requirements, navigational hazards, weather and tidal conditions, and other predictable factors. A well thought out navigation plan might even incorporate certain traffic conditions as to avoid crossing traffic such as scheduled high speed ferry departures, to slow down at a harbor entrance, or to change to a specific speed required by maritime authorities in a channel. If a condition arises where a risk of collision exists as determined by the mariner, the mariner might take action and deviate from the planned route.

The parallel is clear for a USV. Under normal conditions, the USV follows the plan generated by the global planner without much deviation, barring sensing or localization errors. Only when risk of collision exists, as determined by a set of conditions in a collision avoidance algorithm (local planner), does the USV plan an evasive action to reduce or eliminate the risk of collision. Naturally, with regard to collision avoidance, researchers have focused on responsiveness and accuracy as opposed to efficiency for global path planning.



(1) the decision process in a manned ship



(2) the decision process in an autonomous ship

Figure 1-1: In an extensive review of the state-of-the-art collision avoidance methods, Huang et. al collected developments of collision prevention techniques either for manned ships or unmanned ships. Huang et. al used this diagram to contrast the decision process onboard manned ships and unmanned ships and evaluated the methods under this framework. Image from [22].

.

1.2 Path to Autonomy - A Literature Review of Selective Methods

1.2.1 Global Path Planning

Global path planning is a classic research area with deep roots in computer science. With known static obstacles, the general strategy is to first discretize the environment and turn it into a grid. Then the problem of going from point A to point B is thereby transformed into a graph search problem well suited for seminal algorithms

like Dijkstra's, RRT (Rapidly-exploring Random Trees), or the most commonly used A* [32, 16]. Faced with complex variables in a maritime environment, the major limiting factor of these algorithms is speed [32]. Recent developments are either focused on improving these classic methods to make more usable products for the maritime domain, or combining global path planning algorithms with a suitable local path planner.

One example is an attempt to make a generic path more suitable to ships. The A* generated optimal path is not always compatible for a vessel to follow, as there could be many sharp twist and turns. Song et. al sought to increase the practicability of a A* path by a series of three smoothing filters [42]. Post smoothing both the path length and the number of turns in the path compared to traditional A* are reduced, as shown in Figure 1-2. Though capable of avoiding static obstacles, this method is not suitable for dynamic obstacles as it was implemented offline and takes more than 5 times as long as traditional A*.

Other attempts to improve upon the globally planned path involve coupling it with a faster local path planner. Figure 1-3 exemplifies this approach. Chen et. al, propose to use traditional A* as the global planner and use a dynamic window algorithm (DWA) [47], to ensure collision avoidance performance. With this method, even if collision avoidance is required, the USV will always attempt to get back to the global path.

Liu et. al presented yet another example of a unified approach [32]. The authors proposed dynamic obstacle avoidance and path planning problem of USV based on the Ant Colony Algorithm (ACA) and the Clustering Algorithm (CA) to construct an auto-obstacle avoidance method that is suitable for the complicated maritime environment. While this method has a speed advantage, the collision avoidance is not protocol based. So a user does not have any reasonable expectation of how a vehicle will navigate when risk of collision exists.

Despite the trend for combined or hybrid planning algorithms, it appears that general structure does not veer away from that in Figure 1-1. Even when integrated, the literature still operates under the paradigm that the global path is the primary

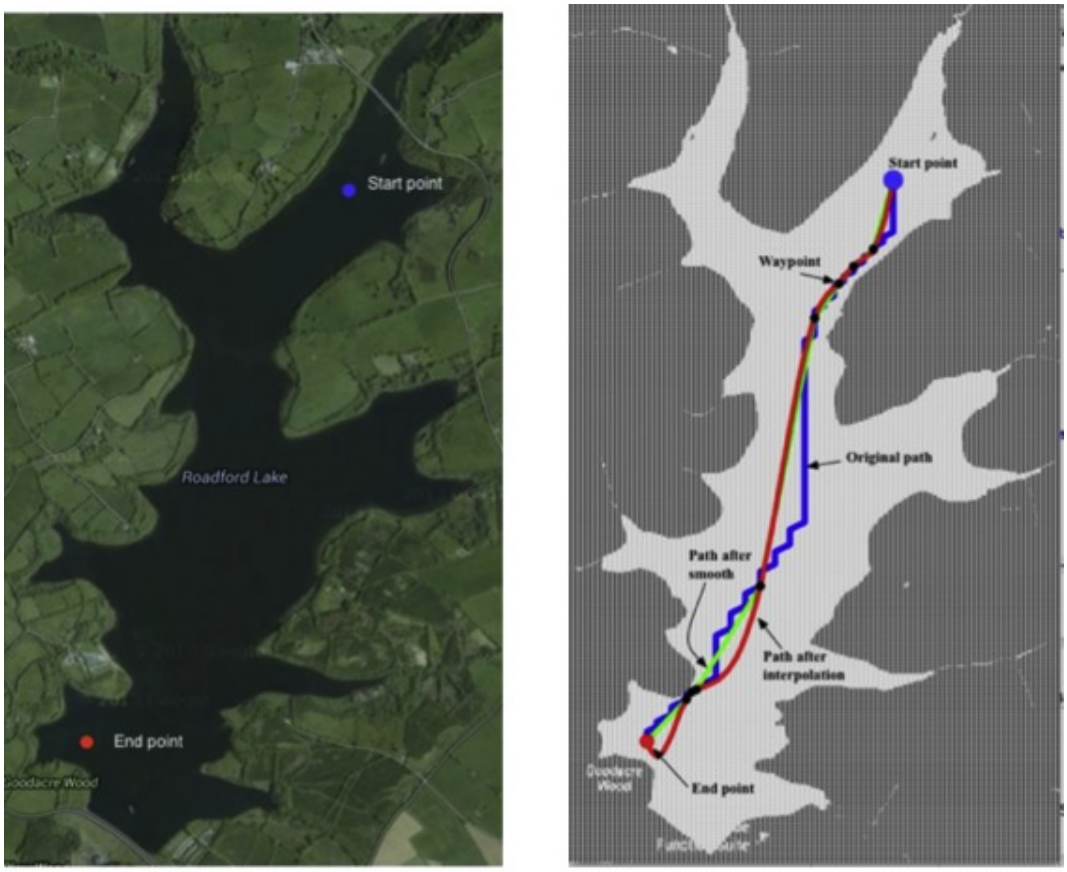


Figure 1-2: Off-line path planning simulation result. (a) The satellite map (b) The binary grid map with trajectories generated using different smoothing techniques. Blue line is original A*, Red line is final smoothed route. Figure from [42].

plan and a local path is reactive. The global path does not make allowance for the local planner unless a risky situation develops. The next section will study the different approaches on deciding when a situation is risky.

1.2.2 Collision Risk Assessment and Resolution

Fundamental to any judgement of the navigational situation is the judgement of risk. For both humans and robots, that judgement might be based on a variety of factors such as TCPA (Time to Closest Point of Approach), DCPA (Distance to Closest Point of Approach), relative bearing, speed, visibility, etc. Given a set of parameters, the judgement belies a risk model that is generally implicit for a human operator,

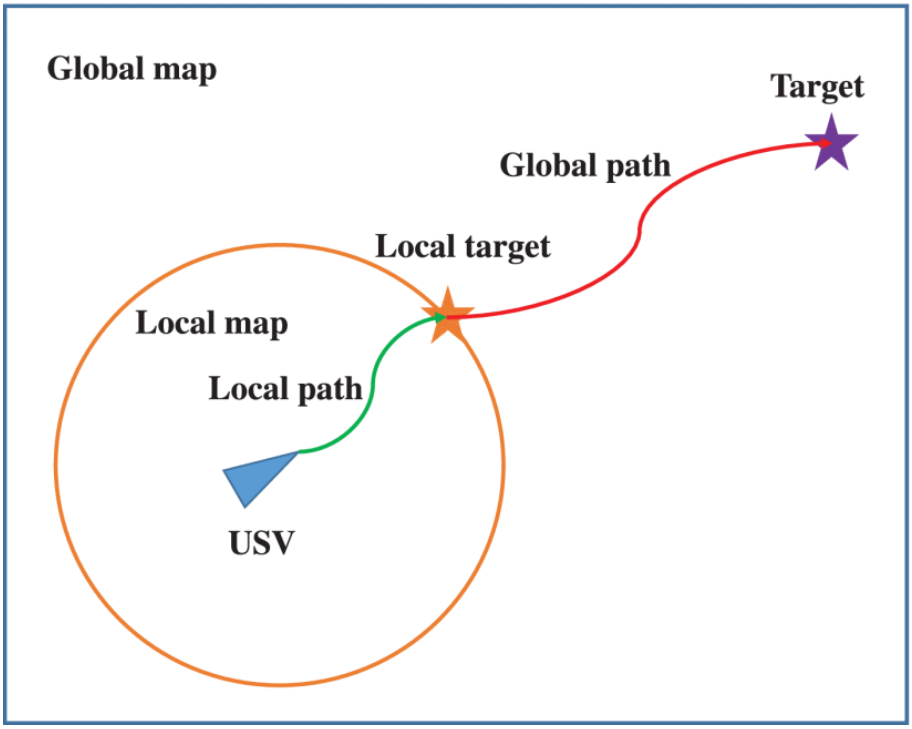


Figure 1-3: Vehicle use local planer within the circle, with the intersection with the global path as the goal. Figure from [16].

mathematically formulated for an autonomous system. The following section will systematically review relevant risk models developed in the past.

The wealth of literature on collision risk assessment can be roughly divided into two camps: ones that formulate probability models to estimate the likelihood of a collision within a geographical area and ones that estimate the probability of a collision for a particular ship given the navigational environment [38]. The first camp takes a frequentist view to risk and these models are generally developed with harbor control or transportation analysis applications in mind. Roboticists with collision avoidance applications in mind evaluate risk from the perspective of a home platform, and develop risk models that fall into the second camp.

Among risk models developed for collision avoidance, the most common features considered are TCPA, DCPA, and relative bearing [38]. But due to the number of feature parameters and the inherently complex marine environment, the consensus on how to evaluate risk ends there. Goerlandt and Montewka (2015) presented a sys-

tematic way to categorize risk model development into eight different categories [19]. Only one category, the *strong realist* models, will be discussed here for its relevance to this research.

Strong Realist

The strong realist models are characterized by [19]:

1. Risk is considered to exist objectively as a physical attribute of a system, and the analysis is presented as an estimate of this underlying true risk.
2. It exclusively relies on data collected from the system or on engineering science model.
3. Expert judgment is not considered a source of evidence.
4. Evidence uncertainty is not considered.
5. Stakeholders are not involved in the process of analysis.
6. There exists a strict separation between facts and non-epistemic values.
7. Contextual risk attributes (such as fear) are not considered
8. There exists a strong relation to established risk decision criteria.

The best example of a strong realist approach is Closest Point of Approach (CPA) based risk assessments, i.e. based on the ships' locations that attain the smallest distance during the process of ships approaching each other. CPA is a intuitive indicator of the ship's collision risk that is commonly used by mariners for navigation and by researchers to define various risk indices. The time and distance to CPA (TCPA and DCPA) measurements give further context as to when and where a potential close encounter will take place. In the open ocean, many mariners have a set CPA that they consider to be the safe distance, and may decide to maneuver in order to maintain it. This is typically accomplished by evaluating the new CPA with respect to candidate maneuvers, then choose one in accordance with COLREGS (the International

Regulations for Preventing Collisions at Sea adopted by the International Maritime Organization(IMO)) [13].

Through case studies and experiments, many navigation experts and authors have concluded that DCPA and TCPA do not fully reflect severity of an encounter [36, 19, 44]. Researchers proposed methods to aggregate TCPA, DCPA, and other selected factors to form a single "risk degree", but these more comprehensive risk measurements do not necessarily lead to obvious mitigation methods that could reduce said risk [21, 33, 41].

An improvement on the one dimensional measurement of CPA, is a concept called the *ship domain* [46]. Ship domain differs from safe distance in the recognition that safe distance is not the same in all directions. Ship domain is a safety region around a ship that allows a navigator to take timely action to avoid any potential collision. As such, ship domain should be kept to be the minimum to have practical value. If the ship domain only encompasses the minimum safety region, then an entrance into ship domain is equivalent to a threat of collision, and allows no time for evasive reaction. If the ship domain is defined too generously, then too many navigational situations might breach it and render the measure useless. Therefore, the size of ship domain is not exact and could be affected by the following factors: vessel shape and size, ship speed and course, regional traffic density, environmental condition, and even ship crew's skill and responsiveness. The simple model would be just a rectangular domain at a set multiple of ship's dimensions [52]. A more geometrically pleasing ellipsoidal ship domain can be constructed weighing heading, speed, and relative motion of target [46], as shown in Figure 1-4. Other approaches incorporate encounter geometry to distinguish head on, crossing, and overtaking situations [51].

Projecting ship domain outwards in a concentric fashion, as shown in Figure 1-5, to intersect predicted contact geometry is a viable way to determine collision risk. The first example, Fuzzy ship domain, can be adapted to various encounter scenarios based on expert judgement. While this risk measurement approach is not a *strong realist* one, it is included because it is a popular approach [39, 34, 1, 26]. Fuzzy logic approaches also offer risk reduction and conflict resolution methods based on

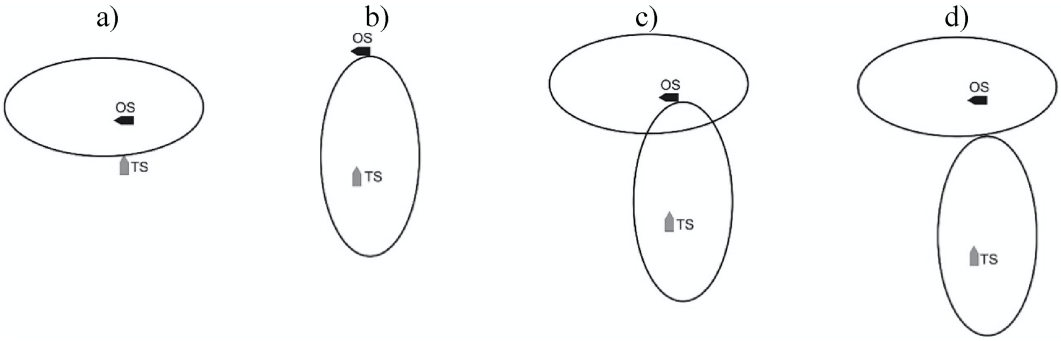
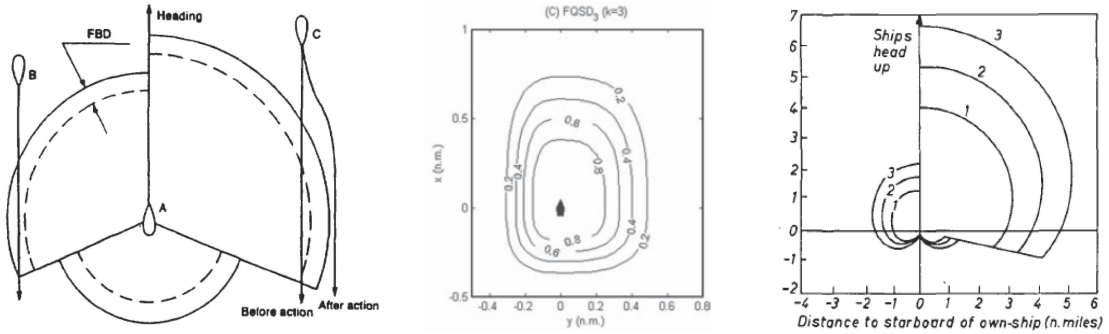


Figure 1-4: Different domain-based safety criteria: a) OS domain is not violated, b) TS domain is not violated, c) neither OS nor TS domain is violated, d) domains do not overlap [46].

evaluation of candidate maneuvers with expert knowledge.



(1) Fuzzy ship domain (2) Spatial collision risk (3) Ship arena

Figure 1-5: Three examples of using ship domain for collision detection or collision risk assignment. Figure from [22].

.

With the knowledge of a danger area such as the ship domain, algorithms can be devised to avoid it. One early example is Artificial Potential Field (APF), shown in a Figure 1-6, a method that generates repulsive or attractive potential fields around objects on the map, then guides the robot through the virtual force generated by these fields [27]. As a path generation method, APF provides a direction of movement, but does not directly provide a collision free path [22]. Further studies have expanded its use to collision avoidance, include modifications to enable partial COLREGS compliance [35].

An extension of a danger area is Velocity Obstacle (VO) and its variations [23, 31, 10]. Instead of looking at the geometric space of relative positions, VO generates

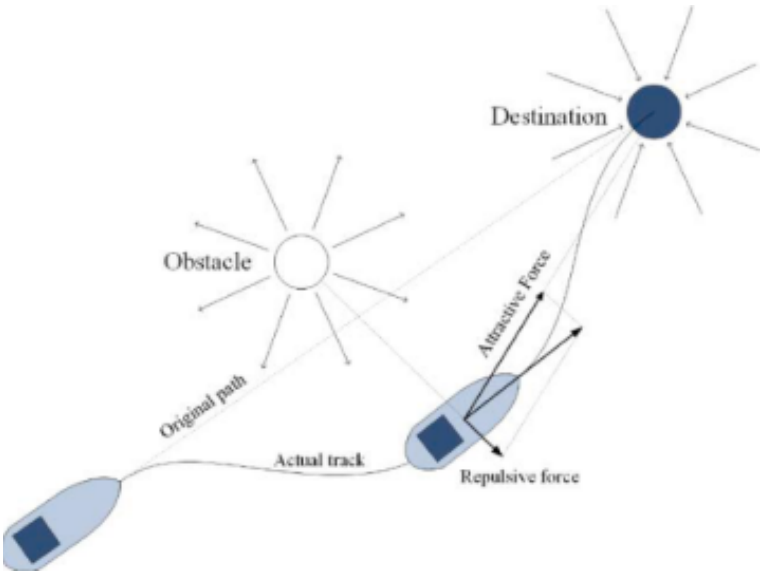


Figure 1-6: Visual representation of a Artificial Potential Field (APF). Obstacles generate a repulsive field and destinations generate an attractive field. Ownship is guided by original path with deviation forced by the resultant field. Figure from [22].

a cone-shaped obstacle in the velocity space, shown in Figure 1-7. After a pre-collision check to determine CPA and relative motion, Kuwata et. al used a rule based algorithm to determine the navigational situation and eliminate maneuvers that violate COLREGS [31].

A generalization of VO is the velocity function proposed by Benjamin [10]. Instead of the binary classification of causing a collision or not, the velocity function allows for assignment of utilities for all maneuvering choices within the domain. Figure 1-8 shows two ships (ownship in white, and contact in grey) meeting on the left and the one dimensional representation of the velocity function of ownship on the right. Candidate maneuvers that cause collisions are effectively ruled out with utility of 0, represented by dark blue. This method can also be coupled with rule based algorithms to achieve partial COLREGS compliance [8, 49, 50].

While not a comprehensive review of all collision methods, this literature review seeks to establish three points: 1) In general, global path planning algorithms of the past are not responsive enough for maritime collision avoidance. 2) Comprehensive and complex risk measurement models exist, but do not always translate to collision avoidance methods. 3) Practical collision avoidance methods only select key ingredi-

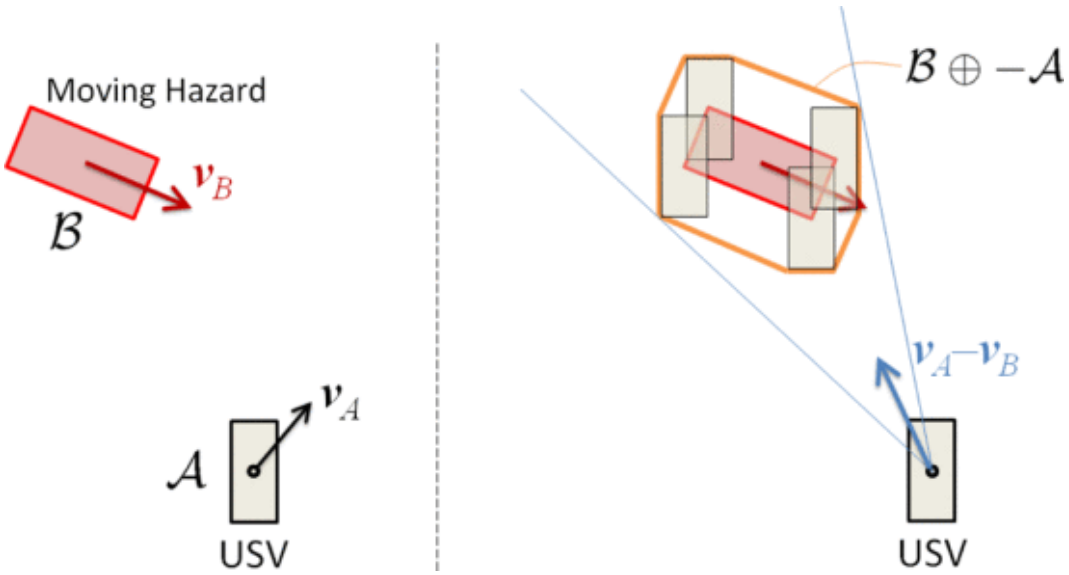


Figure 1-7: Visual representation of a Velocity Obstacle (VO). When object B moves with velocity V_B, the cone around object B indicate the VO. As long as the USV's velocity lies outside the VO, it will not collide with the obstacle, assuming that the velocity vectors are constant over time. Figure from [31].

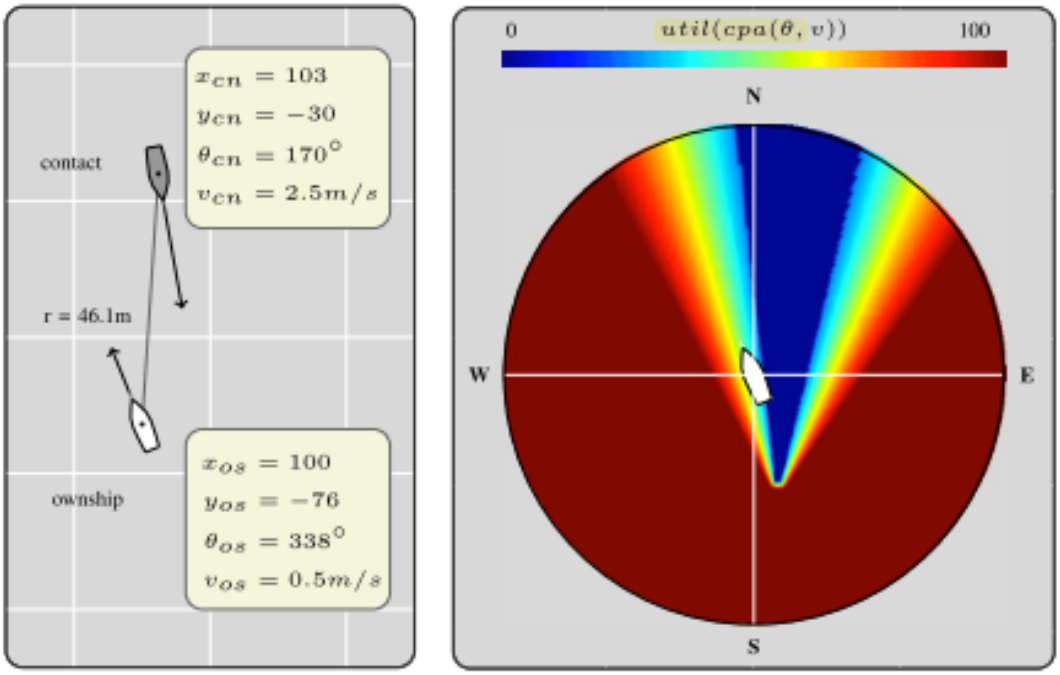


Figure 1-8: A velocity function determined by CPA for all candidate maneuvers in the domain, The function value, or utility, is shown in color gradient with red indicating highest utility and blue the lowest utility. Figure from [10].

ents of collision risk, and trim candidate maneuvers based on a few rules in favor of quick calculation.

1.3 Current Challenges

1.3.1 COLREGS Compliance

Unless new rules are written and adopted for USVs, existing algorithms must be compliant to established protocol for collision avoidance (i.e., COLREGS). Furthermore, the COLREGS protocols these algorithms seek to imitate have clear cut guidance only for single ship-to-ship encounters in canonical geometries, but most real world collisions arise in situations where more than two ships are involved [49]. Therefore, existing algorithms do not yet produce results comparable to experienced human operator in maneuvers that are safe and predictable [50, 30].

1.3.2 Human Interpretation

Part of the challenge is that complex situations require human operator to interpret COLREGS rules. In COLREGS, there are three canonical geometry scenarios for single ship-to-ship encounters: head-on, over-taking, and crossing. For canonical geometries, mariners and autonomous systems can arrive at a straight forward heading/speed solution in partial adherence to COLREGS [49, 25, 31, 35, 39]. Vexing situations arise when more than two ships encounter each other, and the pair-wise geometry between those ships dictate conflicting maneuvers. COLREGS does not give explicit guidance on these situations, but relies on the mariner's "good seamanship". To be fully COLREGS compliant, autonomous systems require an algorithm that looks at the totality of the navigational situation rather than just the pair-wise geometrical relationships.

1.4 Assumptions and Contribution of this thesis

This research takes a strong realist view to model risk based on historical and real-time sensor data without expert knowledge. For an autonomous system, this minimizes the need for operator training but increases the reliability on sensor and platform performance. A detailed list of assumptions is shown below.

1.4.1 Assumptions

1. Reliable and accurate detection and tracking of all contacts is provided to own-ship. They are assumed to have constant heading and speed until updated by new sensor information.

2. Contacts have reliable and accurate detection and tracking of own ship.

1.4.2 Contributions

This thesis makes the following contributions:

1. A novel algorithm that assesses risk of collision for future time steps.

2. A method to adjust the speed of the vessel using this risk assessment to reduce the risk of collision in high density areas.

3. A systematic comparison of collision avoidance behaviors' performance.

1.5 Thesis Overview

Chapter 2 illustrates how the concept of "forehandedness" is an essential trait for good seamanship and proposes general algorithms that emulate this trait. Chapter 3 describes the technical implementation of these algorithms in MOOS-IvP and how it interacts with the existing MOOS-IvP suite. Chapter 4 details the simulation setup and experimental results. Chapter 5 offers conclusions of this work as well as recommendations for future studies.

Chapter 2

General Approach

The mark of a great ship handler is never getting into situations that require great ship handling.

— Fleet Admiral Ernest King

Should USV autonomous operations become common place, frequent manned and unmanned vehicle interactions are inevitable. The difficulty of integrating path planning and collision avoidance algorithms further complicated by the requirement of full COLREGS compliance [30, 32], calls for perhaps another approach. This thesis focuses on the transition between global path planning and reactive collision avoidance and challenges the classical collision prevention information flow paradigm, shown in Figure 2-1.

The famed World War II United States Navy Admiral Ernest King offered the assessment that great ship handlers can recognize potentially risky situations and take early action so that those situations do not arise [43, 17]. This prized trait in the art and science of seamanship is called "forehandedness" [2, 17]. Through the lens of the collision prevention information flow paradigm shown in Figure 2-1, this trait is not neatly captured in any of the stages. Forehandedness certainly requires motion prediction, but by definition must occur before a conflict is detected. To demonstrate the difference between conflict detection and forehandedness, the next section examines a common real world scenario that mariners face.

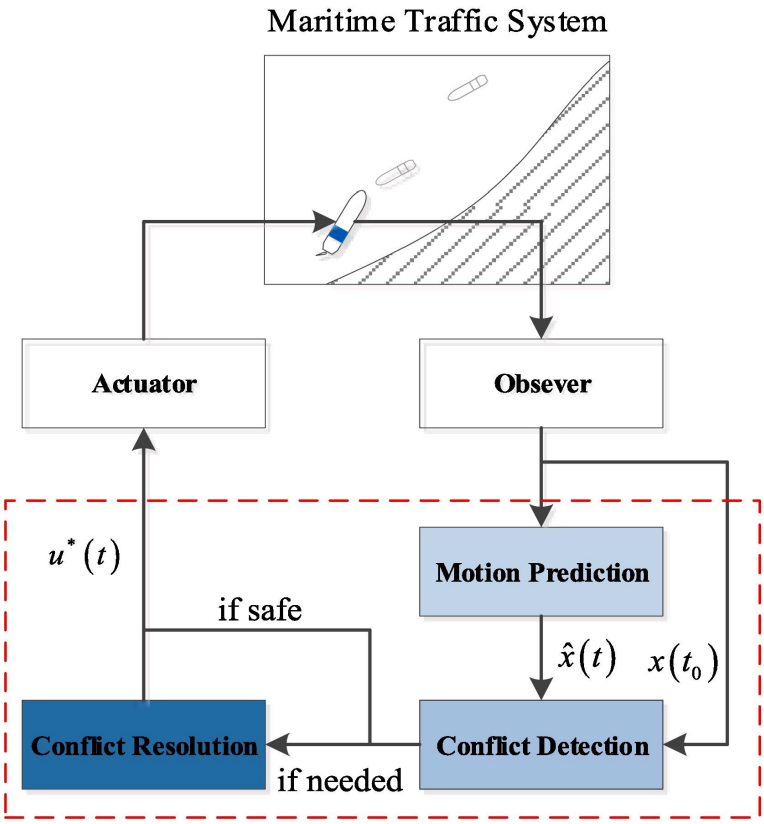


Figure 2-1: Huang et. al abstracted the information flows in both manned and unmanned ships shown above. Collision avoidance literature can be grouped based on the differences of methods in the three key stages: motion prediction, conflict detection, and conflict resolution. Image from [22].

2.1 Approach to a High Traffic Density Area

The Strait of Malacca, a relatively narrow passage which connects the Indian Ocean to the Pacific Ocean, is one of the most traveled maritime choke points in the world. Roughly 25 percent of the world's traded goods flow through this strait, with the annual vessel flow rate growing to nearly 100,000 vessels [12]. To safeguard ships traveling through congested waters like the Strait of Malacca, the IMO approved Traffic Separation Schemes (TSS) that divides traffic into lanes much like a highway on the road. Rule 10 of COLREGS governs vessels operating in or near a TSS, is shown below [13]:

RULE 10: Traffic Separation Schemes (International)

(a) This Rule applies to traffic separation schemes adopted by the Organization and does not relieve any vessel of her obligation under any other rule.

(b) A vessel using a traffic separation scheme shall: (i) proceed in the appropriate traffic lane in the general direction of traffic flow for that lane; (ii) so far as practicable keep clear of a traffic separation line or separation zone; (iii) normally join or leave a traffic lane at the termination of the lane, but when joining or leaving from either side shall do so at as small an angle to the general direction of traffic flow as practicable.

(c) A vessel shall, so far as practicable, avoid crossing traffic lanes but if obliged to do so shall cross on a heading as nearly as practicable at right angles to the general direction of traffic flow.

(d) (i) A vessel shall not use an inshore traffic zone when she can safely use the appropriate traffic lane within the adjacent traffic separation scheme. However, vessels of less than 20 meters in length, sailing vessels and vessels engaged in fishing may use the inshore traffic zone. (ii) Notwithstanding subparagraph (d)(i), a vessel may use an inshore traffic zone when en route to or from a port, offshore installation or structure, pilot station or any other place situated within the inshore traffic zone, or to avoid immediate danger.

(e) A vessel other than a crossing vessel or a vessel joining or leaving a lane shall not normally enter a separation zone or cross a separation line except: (i) in cases of emergency to avoid immediate danger; (ii) to engage in fishing within a separation zone.

(f) A vessel navigating in areas near the terminations of traffic separation schemes shall do so with particular caution.

(g) A vessel shall so far as practicable avoid anchoring in a traffic separation scheme or in areas near its terminations.

(h) A vessel not using a traffic separation scheme shall avoid it by as wide a margin as is practicable.

(i) A vessel engaged in fishing shall not impede the passage of any vessel following a traffic lane.

(j) A vessel of less than 20 meters in length or a sailing vessel shall not impede the safe passage of a power-driven vessel following a traffic lane.

A snapshot of the Strait of Malacca traffic is shown in Figure 2-2 with vessels A through F labeled to illustrate a variety of navigation perspectives.

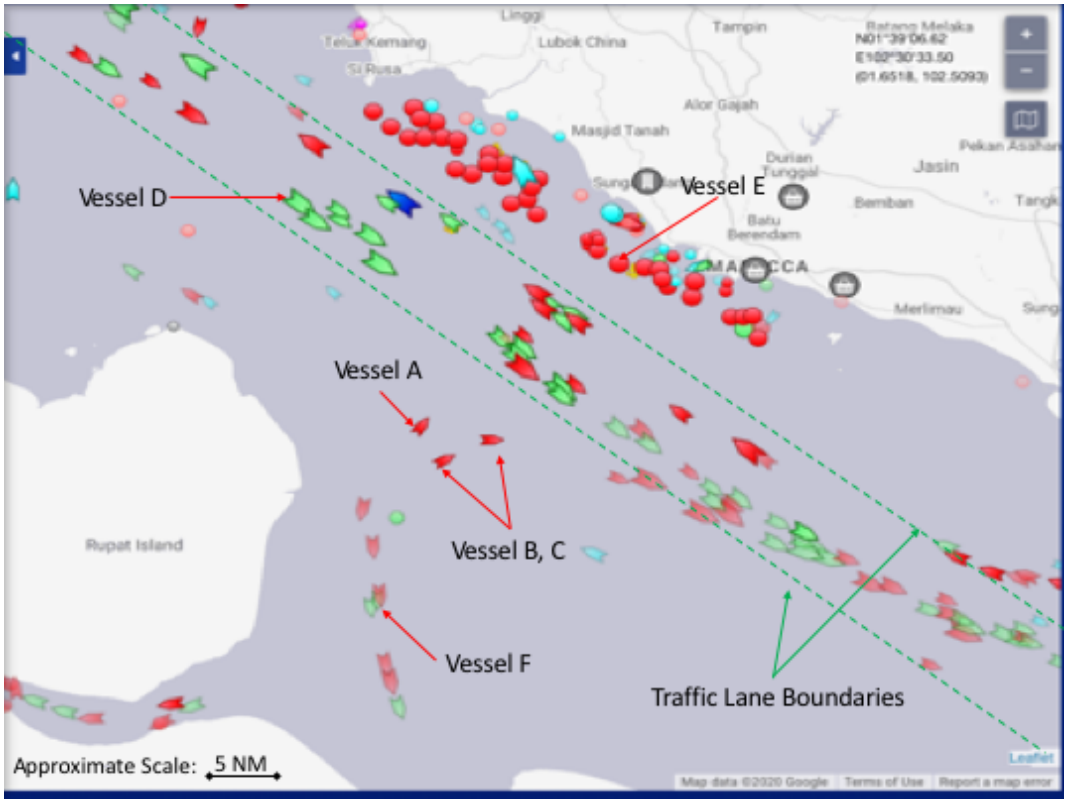


Figure 2-2: A snapshot of marine traffic in the Strait of Malacca (Source: MarineTraffic.com).

The general flow of traffic is lined by the green dotted line, of which vessel D is inside of. While vessel D is in close proximity to a number of vessels, the risk of

collision is small provided that rest of the vessels follow Rule 10 of COLREGS and associated IMO rules specific to this strait [13, 48]. Similarly for vessel E, though in close proximity to a number of vessels, faces a small risk of collision as she and her neighbors are anchored. Vessels B and C seem to be joining the southbound traffic and should aim to enter the TSS at as small of an angle as practical per Rule 10 Section (b)(iii) [13]. Vessel F likely had departed the TSS earlier and is heading elsewhere. The risk of collision for vessel F is small, as she has plenty of maneuvering room to the left and right. Unlike the other vessels mentioned above, vessel A faces a navigational path that is more tortuous. AIS data shows that vessel A's destination is in the vicinity of the Port of Malacca, which requires it to somehow cut across TSS traffic. Per Rule 10 Section (c), she must strive to cross the traffic lanes at a right angle as practicable. A few possible route choices for vessel A is shown in Figure 2-3.

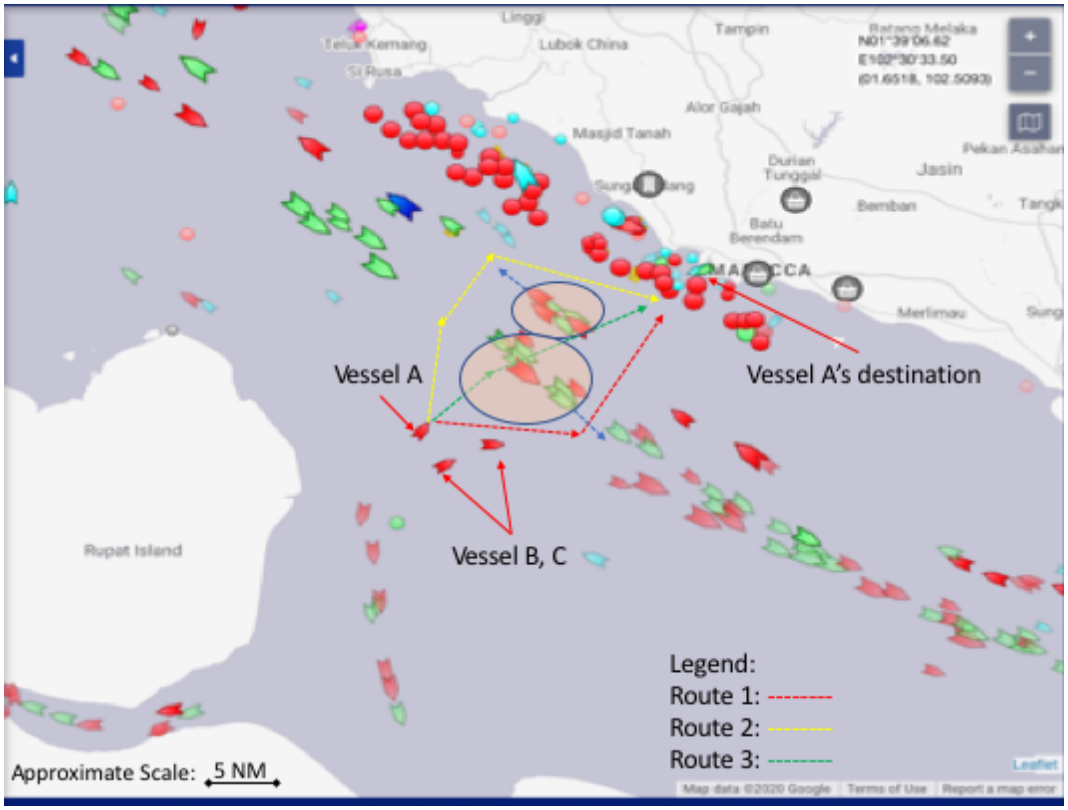


Figure 2-3: Vessel A's possible route choices from presence location to its port of call in Malacca (Source: MarineTraffic.com).

Route 1, denoted in red dashed line in Figure 2-3, is a route that requires great

speed from the vessel. In order to safely cross at close to a right angle, she must beat the southbound traffic, all the while avoiding sideswiping vessels B and C. Route 1's crossing maneuvering is particular dangerous as vessel A must conduct a port turn towards incoming traffic. Since a turn reduces the forward speed of a ship, this temporary slow down would likely decrease the crossing clearance between vessel A and the group of southbound vessels. From a collision avoidance algorithm perspective, Vessel B and vessel C's maneuvering intentions would prove to be challenging for holonomic models to predict, since these models explicitly ignore angular and linear accelerations [24, 46, 11, 22]. A Cooperative Multi-Vessel System (CMVS) might recommend a solution similar to route 1, provided vessels A, B, and C have communicated their intentions and vessel A is speedy enough to overtake southbound traffic before coming into close contact with the northbound ships [14].

A more palpable solution might be route 2, shown in the yellow dashed line in Figure 2-3. Though less direct than route 1, this route offers plenty of sea room for vessel A to cross the TSS. However, she needs to watch for two key factors to ensure safe crossing: 1) the speed of the northbound traffic, and 2) the movements of the group of three boats (cyan colored) near the second waypoint of route 2. This route is a reasonable solution that many DCPA and TCPA based conflict detection algorithms might find safe [18, 22]. Even with a generous ship domain, the available sea room is conducive for collision avoidance algorithms to find a solution similar to route 2 [45]. Overall, route 2 is a safe solution that requires a sharp starboard turn at waypoint 2, a maneuver well within reach of all ships but the heaviest and most restrictive in maneuverability.

On first glance, route 3 is not a viable path for vessel A, since it leads her directly into the southbound traffic. Consider, however, that vessel A decides to substantially slow its approach to the TSS until the southbound traffic (lower blue circle) and northbound traffic (upper blue circle) clear, shown in Figure 2-3. Choosing to slow down and wait requires some foresight and patience, i.e. forehandedness, on the part of the mariner. This type of forward prediction and patience differ from early detection of conflicts, and are not explicitly represented in any of the state of the art

collision avoidance algorithms to date [22]. In other words, forehandedness, an essential and prized trait of good seamanship, the one trait that "separates a good seaman from a merely competent one" [17], has not been modeled. Without forehandedness, collision avoidance algorithms will fall short of the requirement set by COLREGS Rule 8 part (a), shown below:

RULE 8: Action to Avoid Collision (International)

(a) Any action taken to avoid collision shall be taken in accordance with the Rules of this Part and shall, if the circumstances of the case admit, be positive, made in ample time and with due regard to the observance of good seamanship.

(b) Any alteration of course and/or speed to avoid collision shall, if the circumstances of the case admit, be large enough to be readily apparent to another vessel observing visually or by radar; a succession of small alterations of course and/or speed should be avoided.

(c) If there is sufficient sea room, alteration of course alone may be the most effective action to avoid a close-quarters situation provided that it is made in good time, is substantial and does not result in another close- quarters situation.

(d) Action taken to avoid collision with another vessel shall be such as to result in passing at a safe distance. The effectiveness of the action shall be carefully checked until the other vessel is finally past and clear.

(e) If necessary to avoid collision or allow more time to assess the situation, a vessel shall slacken her speed or take all way off by stopping or reversing her means of propulsion.

(f) (i) A vessel which, by any of these rules, is required not to impede the passage or safe passage of another vessel shall, when required by the cir-

cumstances of the case, take early action to allow sufficient sea room for the safe passage of the other vessel. (ii) A vessel required not to impede the passage or safe passage of another vessel is not relieved of this obligation if approaching the other vessel so as to involve risk of collision and shall, when taking action, have full regard to the action which may be required by the rules of this part. (iii) A vessel, the passage of which is not to be impeded remains fully obliged to comply with the rules of this part when the two vessels are approaching one another so as to involve risk of collision.

2.2 Proposed Solution

This thesis proposes a transitional phase, called planned phase, between pre-planned path following and reactive collision avoidance. During the planned phase, the vehicle will reason about future traffic density inside the traffic zone and vary the speed of the vehicle in order to influence the time of arrival to the transition point, as shown in Figure 2-4.

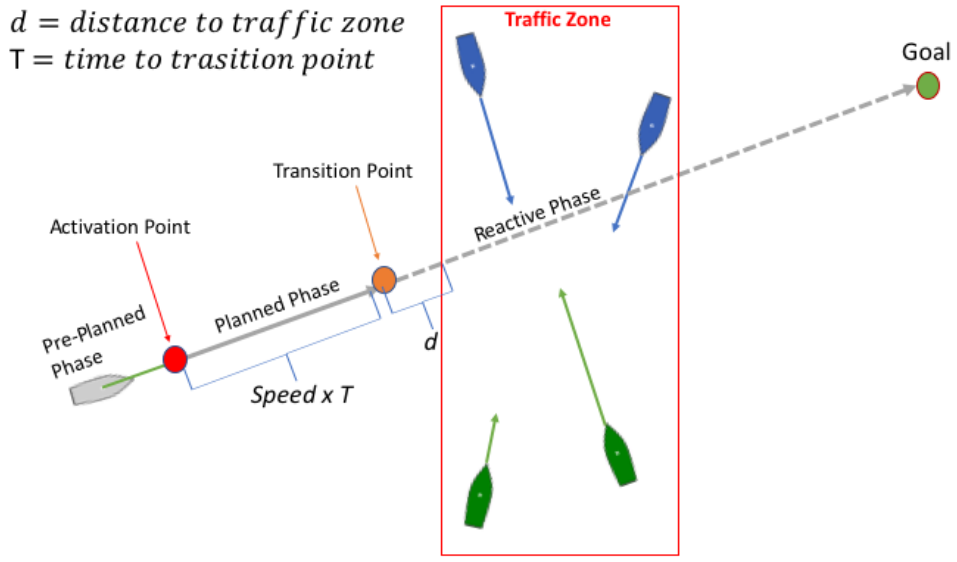


Figure 2-4: Planned approach to traffic zone.

2.2.1 Pre-requisites

The pre-requisites of this approach include the following:

1. A distribution of traffic density so that dense traffic areas have known boundaries, so that they may be approximated by geometric shapes such as polygons.
2. A given set of pre-planned waypoints that traverse through both high and low traffic density areas.
3. Ownship's mission allows it some flexibility in deviating from planned track even in non-emergency.
4. Ownship has access to fairly accurate and regularly updated AIS reports or other means of contact tracking.

This approach does not require that:

1. High density areas are Traffic Separation Schemes with clearly defined traffic patterns.
2. Contacts obey COLREGS.
3. Contacts conduct collision avoidance.

To implement a planned phase, two parameters must be selected, as shown in Figure 2-4. First, the distance d from the transition point to traffic zone. This parameter is set so that before the transition point, ownship is not expected to be maneuvering to avoid collision with ships inside the traffic zone. The second parameter T is the desired time spent in the planned phase. T is primarily selected based on the degree of flexibility ownship has in deviating from its planned track.

2.2.2 Reasoning Process

The overarching algorithm of the planned phase seeks to emulate a mariner who thinks ahead before risk of collision exists. Algorithm 1 conducts continuous density

analysis on vessels inside the traffic zone for all possible speeds within ownship's domain of selectable speeds. Starting at the activation point, ownship no longer blindly follows pre-planned speed. Instead it will choose a speed that both meets the mission requirement and minimizes the maximum density as calculated Algorithm 2.

The key difference between Algorithm 1 and existing collision detection methods is pro-activeness. This algorithm simulates every possible speed choice and considers the worst case scenario each choice might bring ownship into. Popular methods that measure collision risk by DCPA and TCPA, including generalized numerical values such as Collision Risk Index (CRI), trigger consideration of speed and course changes *after* a preset threshold is reached [11, 23, 31, 29, 38, 49]. Moderate realists and moderate constructionist models that value expert judgments import mariners' maneuvering and risk assessment to some extent, but suffer from the scope of expert data collection [20, 36, 28, 40]. These data are collected by asking expert mariners to carefully evaluate a presented navigational situation, generally with just two vessels, and make a maneuvering decision or risk assessment *as is*. What is lacking in these data is the temporal flexibility that allow an expert mariner to select actions that might prevent the presented situation from happening in the first place. In this case, forehandedness is not only not modeled, but implicitly excluded due to the design of experiments.

Algorithm 1 Planned Phase Reasoning

1: **procedure** TRAFFIC($Ownship(O), Contacts(C), Goal(G), TrafficZone(Z)$)
2: $a \leftarrow dist(O, Z)$ ▷ distance from ownship to traffic zone
3: $s \leftarrow speed$ ▷ ship's planned speed
4: **while** $d < a < d + s * T$ **do** ▷ while ship is in planned phase
5: $list \leftarrow DensityCounterOutput$ ▷ calculate ordered pair of speed vs max density
6: $s \leftarrow best(list, speed)$ ▷ select optimal speed that minimizes future density while weighing pre-planned speed
7: **end while**
8: **end procedure**

2.2.3 Density Analysis

The general planned phase reasoning process presented in Algorithm 1 seeks to avoid undesirable and risky situations in the future. The vast literature of collision risk assessment as covered in Section 1.2.2, offers many options to quantify risk [38, 15]. Of these methods, geometric collision probability are methods that work well for ship to ship interactions but difficult to generalize to multi-ship scenarios without arbitrary user inputted weightings [15]. Statistical analysis on accident data including regression or learning not only requires a vast amount of data that are difficult to collect, but might also carries additional computation cost [15]. Due to these concerns, a simplification of what constitutes collision risk is introduced.

Algorithm 2, Future Density Calculation (FDC) takes the naive view that traffic density, as defined by the number of vessels within a preset range limit r, is a proxy measure for the risk of collision. If FDC were to be used as a collision detection tool, it would fail miserably short of that requirement. For FDC does not fully capture the relative dynamic between ownship and contact vessels, nor does it recommend a suitable maneuver to get ownship out of harms way. Instead, FDC is performed before meeting thresholds that trigger reactive collision avoidance algorithms. Its usefulness lies in the fact that it guides ownship out of situations where the potential for risky situation is likely to be higher. When FDC is introduced in the ownship speed consideration in step 6 of Algorithm 1, the effect is a type of vessel behavior that seeks to maximize sea room.

2.2.4 Algorithm Termination

Past the transition point, the vessel will terminate Algorithm 1 and resume its path following behavior. The time of arrival at the transition point now differs from the original plan, so that while going through the traffic lane, the maximum density is less than or equal to what the pre-planned track would have resulted.

Algorithm 2 Future Density Calculation

procedure DENSITY COUNTER($Ownship(O), Contacts(C), Goal(G)$)
  $M = empty$ ▷ start with an empty map of speed vs density
  $s = 0.1$ ▷ starting simulation speed of 0.1
  for $s < maxspeed$ **do**
    $d_{peak} = 0$ ▷ peak density count
    $t = 1$ ▷ simulation step
    $r \leftarrow rangelimit$ ▷ preset range limit
    $l \leftarrow dist(O, G)/(s \times step)$ ▷ number of steps to reach goal
    for $t < l$ **do**
      $d = 0$ ▷ density count
      $O \leftarrow O(t)$ ▷ update ownship position
      for $C_i \in C$ **do** ▷ check contacts in O one by one
        $C_i \leftarrow C_i(t)$ ▷ update with contact position
        if $dist(C_i, O) < r$ **then** ▷ check distance between with contact and ownship
          $d \leftarrow d + 1$
        end if
      end for
      if $d_{peak} < d$ **then**
        $d_{peak} \leftarrow d$ ▷ track peak density
      end if
      $t \leftarrow t + 1$
    end for
    $M_s \leftarrow d_{peak}$
    $s \leftarrow s + 0.1$
  end for
  return M ▷ return the complete map
end procedure

Chapter 3

Methodology and Technical Analysis

The planned phase described in Section 2.2 is implemented with the Mission-Oriented Operating Suite with Interval Programming (MOOS-IvP) through the pTrafficDensity application, the DensityCounter class, and the DensityCount Behavior. This chapter gives the technical overview of MOOS-IvP and how the new applications built for the planned phase integrate with the high level autonomy behaviors already built in MOOS-IvP [6].

3.1 MOOS-IvP Introduction

3.1.1 Background

The brief history of MOOS-IvP is summarized in the "An Overview of MOOS-IvP and a Users Guide to the IvP Helm" as follows [6]:

"MOOS was written by Paul Newman in 2001 to support operations with autonomous marine vehicles in the MIT Ocean Engineering and the MIT Sea Grant programs, funded by the Office of Naval Research. At the time Newman was a post-doc working with John Leonard and has since joined the faculty of the Mobile Robotics Group at Oxford University. MOOS continues to be developed and maintained by Newman at Oxford and the most current version can be found at his web site. The MOOS software available in the MOOS-IvP project includes a snapshot of the MOOS

code distributed from Oxford. The IvP Helm was developed in 2004 for autonomous control on unmanned marine surface craft, and later underwater platforms. It was written by Mike Benjamin as a post-doc working with John Leonard, and as a research scientist for the Naval Undersea Warfare Center in Newport Rhode Island. The IvP Helm is a single MOOS process that uses multi-objective optimization to implement behavior coordination."

Since 2004, MOOS-IvP has been implemented on more than 30 types of autonomous platforms including Unmanned Surface Vehicles, Unmanned Underwater Vehicles, as well as Unmanned Ground Vehicles [3]. Some of these implementations were explicitly cited in more than 100 publications, while others are complemented with proprietary software and packaged into products by companies [4]. The likely reason for the wide adaptation of this software are twofold: its open source nature and its backseat driver design philosophy [7]. This open source project consists more than 100,000 lines of C++ code developed over 40-work years, and offers a suite of applications and behaviors [3]. These applications and behaviors together support a capable autonomy system out of the box. MOOS-IvP further allows the separation between vehicle autonomy and vehicle control, so that a user can retain desired autonomous behavior even with platform and hardware changes.

3.1.2 Architecture

In the MOOS environment, applications are distinct processes that communicate with each other through a single publish-subscribe database application called the MOOSDB. When implemented, shoreside station and vehicles will each have their own MOOS community consisting of a single MOOSDB and applications of choice. MOOS also enables cross community messaging, which can be configured to mirror realistic shore station monitoring capability and inter-vehicle communication scheme. Furthermore, each vehicle's onboard MOOS community can be configured to perform a variety of operations from sending command signals to the propulsion system to autonomous collision avoidance.

The IvP helm is itself a MOOS application, whose core function is to produce a

single output heading, speed, and depth (when required) command. This output is obtained by the IvP solver, using a multi-objective optimization problem solving technique called Interval Programming (IvP) [5]. Each active MOOS behavior produces a piecewise linearly-defined objective function that reflects that particular behavior's desired vehicle maneuvers. The solver then reconciles these objective function inputs using user-defined priority weights.

3.1.3 More on the IvP Solver

Mathematically, each interval programming problem consists of a collection of objective functions $f_i(x_1, x_2, ..., x_n)$ where x_i are decision variables, i.e., speed, heading, or depth. For a problem with k objective functions, the IvP solver finds the optimal maneuver $\vec{x^*}$ by solving the problem:

$$\vec{x^*} = argmax_{(\vec{x})} \sum_{i=1}^{k} (w_i \times f_i(\vec{x}))$$

where $f_i(\vec{x})$ is the objective function of the i^{th} active behavior with associated priority weight w_i.

The conditions under which a behavior is active, as well as the associated priority weight for its objective function, are both user-defined to meet specific mission needs. In a sense, each MOOS behavior is a "self-contained mini-expert systems dedicated to a particular aspect of overall vehicle autonomy" [5]. With the understanding of each behavior's capability, operators consider mission requirements and devise modes of operations for the vehicle. Each mode of operation selects a set of these behaviors. Some fundamental behaviors that come with the MOOS-IvP package include the Waypoint behavior, Loiter behavior, StationKeep behavior, and AvdColregs behavior. Two of the behaviors central to this research are the Waypoint behavior and AvdColregs behavior.

3.1.4 The Waypoint Behavior

The basic function of the Waypoint behavior is to command the vehicle to follow a set of specified waypoints in the x-y plane. The primary parameter is the set of ordered waypoints. These waypoints could be a pre-planned navigation plan entered by the operator or supplied dynamically by an online global path planner. The basic idea is shown in Figure 3-1.

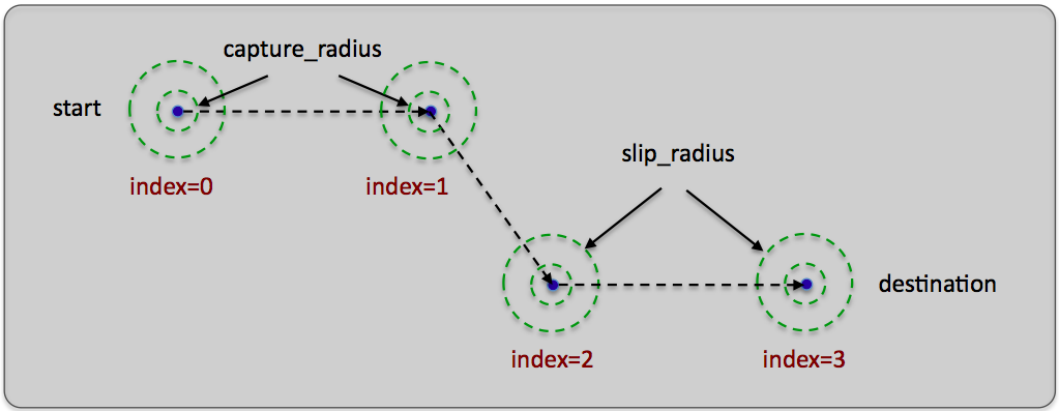


Figure 3-1: The Waypoint behavior basic purpose is to traverse a set of waypoints. A capture radius is specified to define what it means to have achieved a waypoint, and a non-monotonic radius is specified to define what it means to be "close enough" should progress toward the waypoint be noted to degrade. Figure from [6].

Other key parameters are the capture and slip radius around each waypoint that determine what it means to have met the conditions for moving on to the next waypoint. If a vehicle closes the waypoint to a distance less than the capture radius, it is considered to have arrived at that waypoint. The slip radius is set so that if the distance between the vehicle and the waypoint becomes non-monotonic, i.e. goes from closing to opening, the waypoint is considered arrived. This consideration is given to prevent the vehicle from circling back to a waypoint it had just passed outside the capture radius but within the slip radius.

The final discussion point is the objective function produced by the waypoint behavior. Figure 3-2 shows the navigational situation on the left and the top view of the waypoint behavior objective function on the right. The value of the objective function, also called a utility function, is indicated by the color ranging from blue to

red, where the smaller color wavelength corresponds to smaller utility value. In this example, the dark red peak of the objective function is overlaid on the decision space so that its angle is the desired heading and its magnitude is the desired speed. The orange and yellow color surrounding the peak indicate that, if an objective function produced by another behavior is involved in the helm decision, there is flexibility in course and speed change. Furthermore, the waypoint behavior allows the user to adjust where a course change is more preferable than a speed change by modifying the shape of the objective function. For this research, the waypoint behavior is set such that there is no preference between speed or course change.

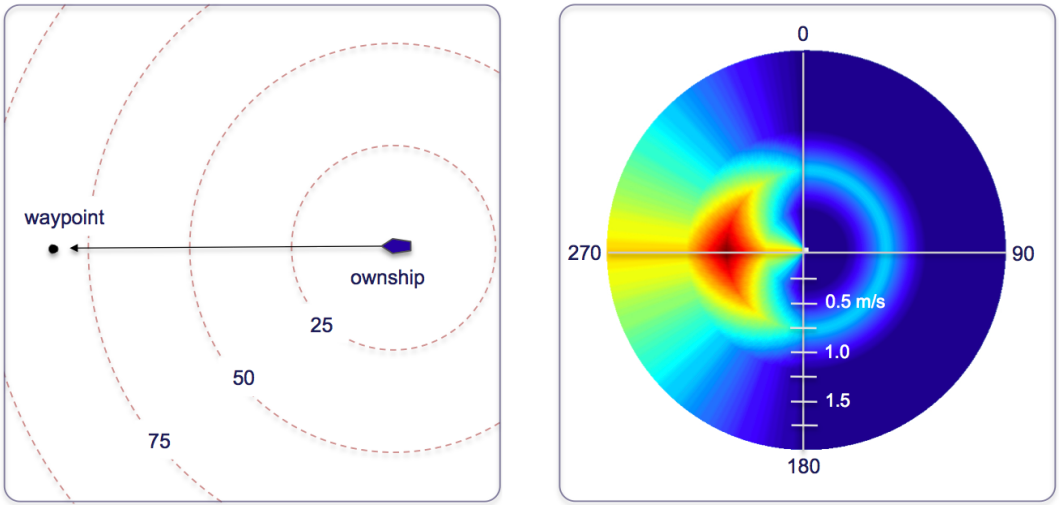


Figure 3-2: The objective function produced by the waypoint behavior is defined over possible heading and speed values. Depicted here is an objective function favoring maneuvers to a waypoint 270 degrees from the current vehicle position and favoring speeds closer to the mid-range of capable vehicle speeds. Higher speeds are represented farther radially out from the center. Figure from [6].

3.1.5 The AvdColregs Behavior

The AvdColregs behavior is designed to produce IvP objective functions that avoid collisions (and near collisions) with another specified vehicle, based on the protocol found in COLREGS [6]. Like other IvP functions, those produced by this behavior are defined over the domain of possible heading and speed choices. The utility assigned to a point in this domain (a heading-speed pair) depends on the relative positions and

trajectories between ownship and a given contact. The priority weight range is user defined, but varies as a function of distance to the contact. This feature ensures that ownship does not immediately disregard its mission whenever a collision avoidance maneuver is been evaluated.

In this research, the AvdColregs behavior is configured so that a new instance is dynamically spawned upon demand as contacts present themselves, typically in the form of a posting to the vehicle's MOOSDB [6]. In the vehicle's MOOS community, the pBasicContactMgr application deals with information about other known vehicles in its vicinity and post alerts as conditions are met. At IvP Helm start-up, a temporary instance of AvdColregs behavior will spawn, post alert requests to pBasicContactMgr, then close out. From then on, the AvdColregs behavior only becomes active when the preset conditions are met.

Configuration parameters for the AvdColregs behavior that are of particular importance for this research are the following [6]:

- **completed_dist**: Range to contact outside of which the behavior completes
- **max_util_cpa_dist**: The distance (in meters) between ownship and the contact at the closest point of approach (CPA) for a candidate maneuver, above which the behavior treats the distance as having the maximum utility, shown in Figure 3-3
- **min_util_cpa_dist**: The distance (in meters) between ownship and the contact at the closest point of approach (CPA) for a candidate maneuver, below which the behavior treats the distance as it would an actual collision between the two vehicles, shown in Figure 3-3
- **pwt**: The priority weight of the behavior
- **pwt_grade**: Grade of priority growth as the contact moves from the pwt_outer_dist to the pwt_inner_dist
- **pwt_inner_dist**: Range to contact within which the behavior has maximum priority weight, shown in Figure 3-4

- **pwt_outer_dist**: Range to contact outside which the behavior has zero priority weight, shown in Figure 3-4. This is also the distance at which a new instance of the AvdColregs behavior will spawn

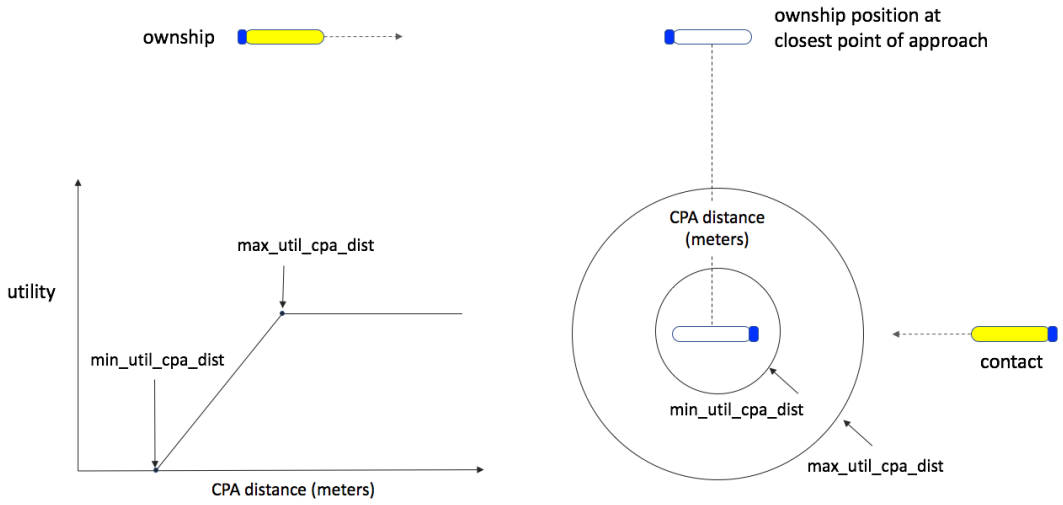


Figure 3-3: The min_util_cpa_dist is used when applying a utility metric to a calculated closest point of approach (CPA) for a candidate maneuver. A CPA less than or equal to the min_util_cpa_dist is treated as an actual collision with the lowest utility rating. Figure from [6].

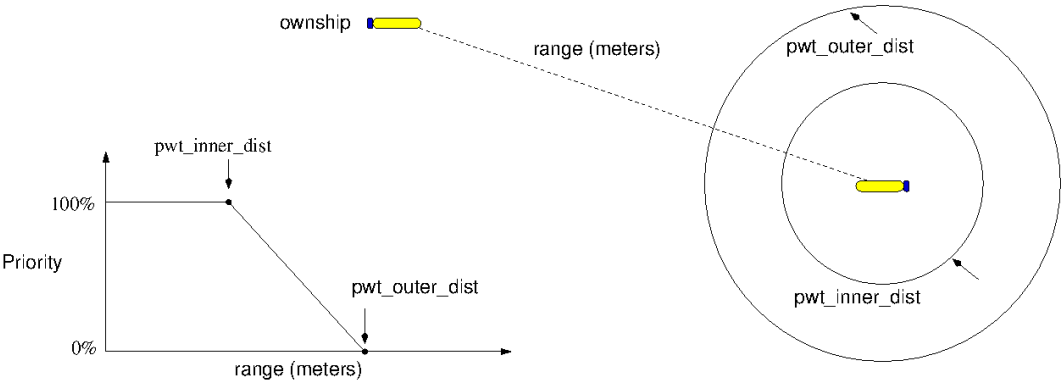


Figure 3-4: The range between the two vehicles affects whether the behavior is active and with what priority weight. Beyond the range specified by pwt_outer_dist, the behavior is not active. Within the range of pwt_inner_dist, the behavior is active with 100 percent of its configured priority weight. Figure from [6].

3.1.6 Path Planning and Collision Avoidance in MOOS-IvP

In summary, the Waypoint behavior offers a way for the vehicle to execute the solution produced by a global path planner, while the AvdColregs behavior offers a reactive collision avoidance algorithm that is partially COLREGS compliant. In fact, the AvdColregs behavior makes no claim to be fully COLREGS compliant, and only seeks to pursue maneuvers that observe COLREGS rules 13-17, a set of rules that governs one-on-one vessel interaction in canonical geometry, i.e. head-on, crossing, and overtaking [8, 49]. Woerner [49] demonstrated both the safety and efficiency of the AvdColregs behavior with close to 750,000 simulated vehicle interactions, further validated with over 100 hours of on-water testing. Unless a robust and standardized autonomous collision avoidance evaluation process is developed and becomes commonly accepted, it is difficult to compare AvdColregs behavior to other methods presented in Section 1.2.3 [50]. Nor is such comparison required. Rather, this research seeks to demonstrate the reduction in risk through the implementation of a planned phase, agnostic to the path planning and collision avoidance algorithms chosen.

3.2 Implementing the Planned Phase

The general relationship between applications developed for the planned phase and key existing applications inside a MOOS community is shown in Figure 3-5. There are other applications and behaviors required to form a full MOOS community that enable a vehicle's autonomy capabilities, though not shown. They will be discussed when necessary. Applications that facilitate shore station monitoring and results analysis that do not interact with the planned phase will be discussed in Chapter 4. Now turning to the first piece of the implementation: the pTrafficDensity application.

3.2.1 pTrafficDensity Application

The pTrafficDensity application interfaces with MOOSDB by subscribing to a number of MOOS variables (upper case by convention): ownship coordinates (NAV_X and

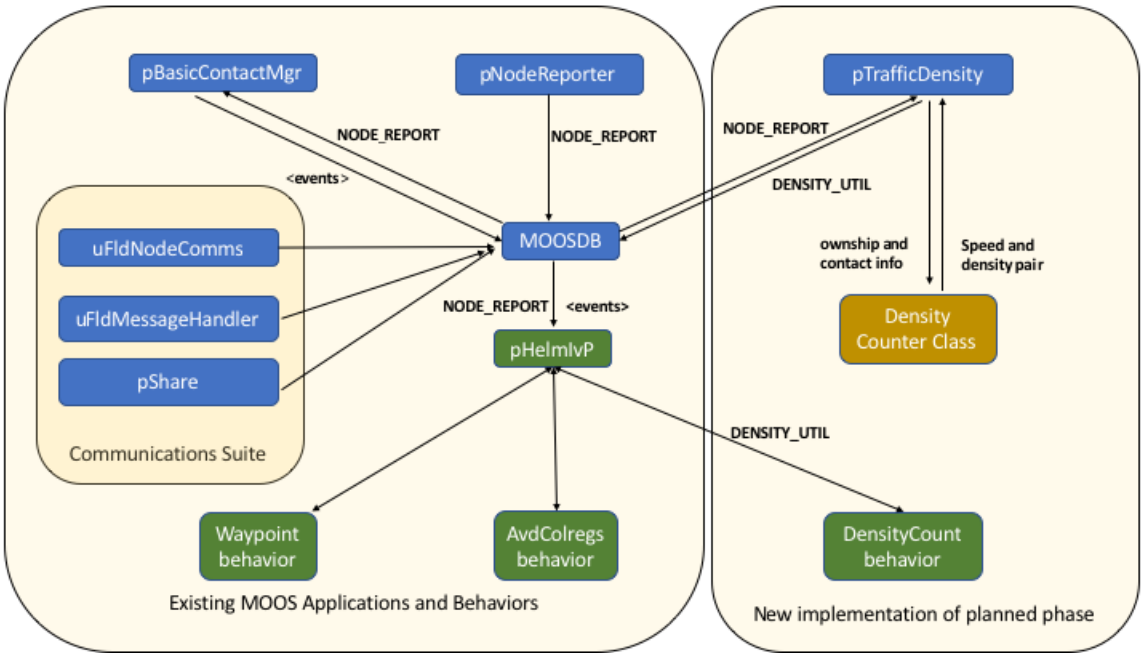


Figure 3-5: The application topology of the vehicle's MOOS community shown here shows how existing applications interact with new ones developed for this research.

NAV_Y), ownship speed (NAV_SPEED), and ownship heading(NAV_HEADING), as well as the NODE_REPORT variable. Node reports are generated on each vehicle by the pNodeReporter application. During every application iteration, pNodeReporter gathers local platform information and navigation data and generates an AIS like report and posts it to the MOOS variable NODE_REPORT_LOCAL [6]. The communications suite, including pShare, uFldNodeComms, and uFldMessage, handle the processing and sharing of NODE_REPORT as shown in 3-6.

Each NODE_REPORT message contains vital information for pTrafficDensity to gain situational awareness of the navigational situation. An example NODE_REPORT string is a comma-separated list of key-value pairs:

```
NODE_REPORT = NAME=abe, X=-100, Y=-75, SPD=2, HDG=179, DEP=0,
LAT=43.82461041,LON=-70.33162829, TYPE=KAYAK, GROUP=contact,
MODE=MODE@ACTIVE:LOITERING, YAW=1.5707963, TIME=23919111020.26,
LENGTH=4
```

As pTrafficDensity receives NODE_REPORT messages from nearby vessels, the application starts to paint a navigation picture. At the same time, pTrafficDensity

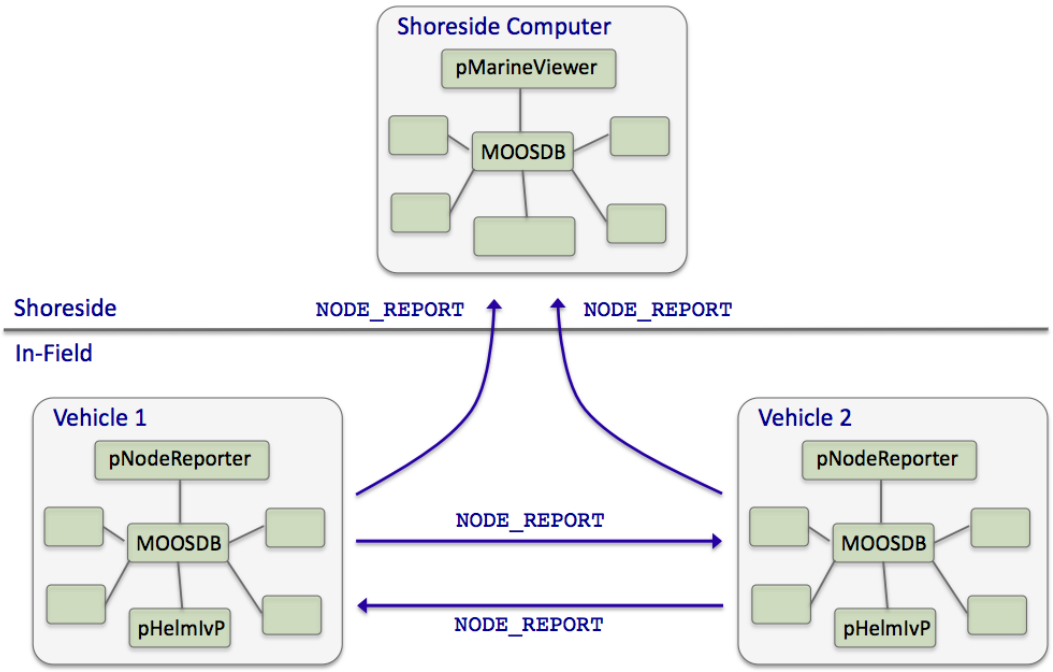


Figure 3-6: Typical pNodeReporter usage: The pNodeReporter application is typically used with pShare or acoustic modems to share node summaries between vehicles and to a shoreside command-and-control GUI. Figure from [6].

also checks MOOSDB for the location of the next vehicle waypoint and stores it as the goal. At each iteration, the application creates an instance of the DensityCounter class using the most up-to-date location, speed, and heading information regarding ownship and contact, as well as the goal location. Once the necessary information for the DensityCounter class is populated, pTrafficDensity uses built-in functions from the class to obtain a vector of speed to density pairs. Each pair represents the *maximum* density the vehicle will encounter if it proceeds at that particular speed from the vehicle's current location all the way to the goal.

The pTrafficDensity application produces two MOOS postings to MOOSDB, DENSITY_COUNT and DENSITY_UTIL. DENSITY_COUNT is simply speed and density vector pairs repackaged into a string of comma separated key-value pairs:

```
DENSITY_COUNT = " 0.10000:3, 0.20000:3, 0.30000:3, 0.40000:3,
0.50000:3, 0.60000:3, 0.70000:3, 0.80000:3, 0.90000:3, 1.00000:3,
1.10000:3, 1.20000:3, 1.30000:3, 1.40000:3, 1.50000:3, 1.60000:3,
1.70000:3, 1.80000:3, 1.90000:3, 2.00000:4, 2.10000:4, 2.20000:4,
```

```
2.30000:4, 2.40000:4, 2.50000:4, 2.60000:3, 2.70000:3, 2.80000:3,
2.90000:3, 3.00000:3, 3.10000:3, 3.20000:3, 3.30000:3, 3.40000:3,
3.50000:3, 3.60000:3, 3.70000:3, 3.80000:3, 3.90000:3, 4.00000:3,
4.10000:3, 4.20000:3, 4.30000:3, 4.40000:2, 4.50000:2, 4.60000:2,
4.70000:2, 4.80000:2, 4.90000:2, 5.00000:2".
```

DENSITY_UTIL is a comma separated string of key-value pairs of speed to corresponding utility. The values are also produced with built-in functions of the DensityCounter. The intended consumer for this variable is the IvP Helm and the DensityCount behavior. An example DENSITY_UTIL posting is shown below:

```
DENSITY_UTIL = "0.1:40, 0.2:40, 0.3:40, 0.4:40, 0.5:40,
0.6:40, 0.7:40, 0.8:40, 0.9:40, 1:40, 1.1:40, 1.2:40, 1.3:40,
1.4:40, 1.5:40, 1.6:40, 1.7:40, 1.8:40, 1.9:40, 2:40, 2.1:40,
2.2:20, 2.3:20, 2.4:20, 2.5:20, 2.6:20, 2.7:40, 2.8:40, 2.9:40,
3:40, 3.1:40, 3.2:40, 3.3:40, 3.4:40, 3.5:40, 3.6:40, 3.7:40,
3.8:40, 3.9:40, 4:40, 4.1:40, 4.2:40, 4.3:40, 4.4:60, 4.5:60,
4.6:60, 4.7:60, 4.8:60, 4.9:60, 5:60".
```

While DENSITY_COUNT and DENSITY_UTIL are the main products of the pTrafficDensity application, it is also configured to give the user monitoring ability through Appcasting. Appcasting is a MOOS-IvP built-in capability that is designed to make it easier to see application terminal output. This includes application specific status messages, configuration and run-time warnings, and notable events [6]. As shown in Figure 3-7, pTrafficDensity is configured to give the user a status report of where it thinks the vehicle is, what contacts it is tracking, as well as the results of peak density calculation. Near the very top of pTrafficDensity's Appcast is information on configuration variables such as the range limit and time step size.

The Range Limit c is a configuration parameter that determines how close a contact needs to be, to be considered within range. If c is set too big, it is conceivable that every speed choice will result in all vehicles being within range thus making this an unusable measure. If c is set too small, then only the most extreme encounters are taken into account. This is also undesirable because the collision avoidance algorithm

is supposed to avert those close encounters. In a way, c represent a caution that an experienced mariner develops, and expresses the vehicle's risk tolerance for proximity to other vehicles.

```
============================================================
pTrafficDensity nelson                                0/0(420)
============================================================
------------------------------------------------------------
Range limit set to 50 meters
Step Size is 1 seconds
------------------------------------------------------------
From App, own ship x, y, hdg, spd: 4.54092,-62.67868 , 92.86898,4.86
Goal location is: x = 270, y = -75
------------------------------------------------------------
From Class: own ship x, y, hdg, spd: 4.54092, -62.67868, 92.86898,4.86 range limit: 50
------------------------------------------------------------
Vname     Contact X     Contact Y     Heading     Speed
-----     ---------     ---------     -------     -----
abe           47.68      -155.04           6       1.2
ben           93.94      -131.32         178      3.75
cal           46.86        13.96      176.01       1.2
deb           77.43        20.17      176.03       1.2
------------------------------------------------------------
nelson's speed choices
------------------------------------------------------------
Speed     Peak Density     Closest     min CPA
-----     ------------     -------     --------
0.1                  2     deb         41.26269
0.2                  2     deb         34.65329
0.3                  2     deb         27.85534
0.4                  3     deb         21.02703
0.5                  3     deb         14.09893
0.6                  3     deb         6.81873
0.7                  3     deb         0.43264
0.8                  3     deb         4.22821
0.9                  3     deb         9.64496
1                    3     deb         5.44099
1.1                  3     deb         0.40886
1.2                  3     deb         5.78109
1.3                  3     deb         10.71961
1.4                  3     deb         15.24265
1.5                  3     deb         19.37965
1.6                  3     deb         23.16529
1.7                  3     deb         26.63836
1.8                  3     deb         29.78724
1.9                  3     deb         32.70176
2                    2     deb         35.35549
2.1                  2     deb         37.79775
2.2                  2     deb         40.05001
2.3                  1     deb         42.13201
2.4                  1     deb         44.05693
2.5                  1     deb         45.82049
2.6                  1     deb         47.48217
2.7                  1     deb         48.9991
2.8                  0     deb         50
2.9                  0     deb         50
3                    0     deb         50
```

Figure 3-7: A snapshot of pTrafficDensity's Appcast. MOOS supports Appcasting to give user run-time feedback.

The time step size, t, is the simulation step to be used by the DensityCounter class, which is set at 1 second. When DensityCounter performs simulation of the future navigational situation, ownship and contacts will update their locations by their respective velocity vector multiplied by t. The detailed steps of this simulation are discussed in the next section.

3.2.2 DensityCounter Class

The DensityCounter class is a C++ class that stores various ownship and contact information and performs calculations with regard to traffic densities. Each incident of the DensityCounter class is created by the pTrafficDensity Application.

Required Inputs

The Density counter class requires at minimum the following information:

- **ownship**: x and y coordinates, speed (optional), heading (optional)
- **Contact Ship**: x and y coordinates, heading, speed
- **Goal Point**: x and y coordinates
- **Range Limit c**: the distance at which a vessel is considered in range (configuration variable)
- **Max Speed S**: the maximum speed that should be considered in simulation (configuration variable)
- **Time Step t**: the temporal increment used in simulation (configuration variable).

Range limit, max speed, and time step are configuration variables that can be set according to ownship's characteristics. A highly maneuverable ship might want to increase the range limit to allow ample time for action. The maximum speed should not be the physical maximum speed that the ship is capable of, but rather a speed that the ship is willing to entertain while in a waypoint following mode. And finally, a high-speed ship might want to reduce the simulation time step to get results with higher spatial resolution.

Density Calculations

The core goal of the DensityCounter class is to use the required inputs to determine what will be the peak density for a given ownship speed. This class implements

Algorithm 2 by performing sequential simulation at the configured time step as shown in Figure 3-8. For each time step, ownship and the contacts move forward by the distance and direction denoted by the solid arrow. After the ranges d1, d2, d3, and d4 are measured, these ranges are compared to the preset range limit c. If for example, d1 and d2 are less than c, then the class will record a density of 2 for that time step. As the simulation moves forward in time, the density is recorded until the maximum density is found for that speed. This process is repeated for speeds 0.1 to max speed S at 0.1 intervals until the ship reaches the goal. Once each speed has a single maximum density associated with it, the simulation ends.

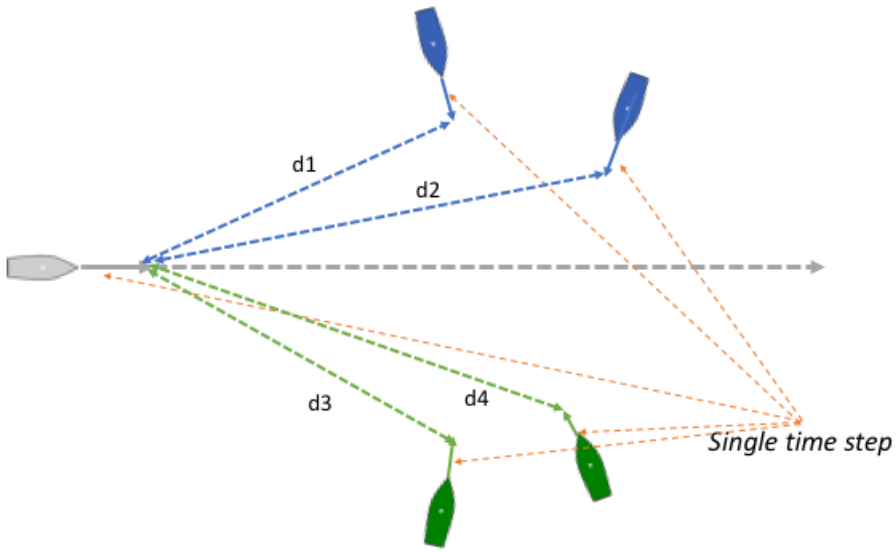


Figure 3-8: Single time step simulation forward illustrated. Key step of Algorithm 2.

Utility Calculations

Figure 3-9 demonstrates a naive version of utility calculation. If, out of four contacts being tracked, three are in range, then the maximum utility of 100 is reduced by 20 times three. In the worst case scenario where all four ship is in range, the resulting utility will be 20. The choice to subtract the maximum utility by 20 for each in-range contact is scaled based on the maximum expected density of 4. If the maximum

number of contacts were higher, the utility calculation can be similarly scaled so that it is inversely correlated to the number of contacts in range, while fall between 1 and 100.

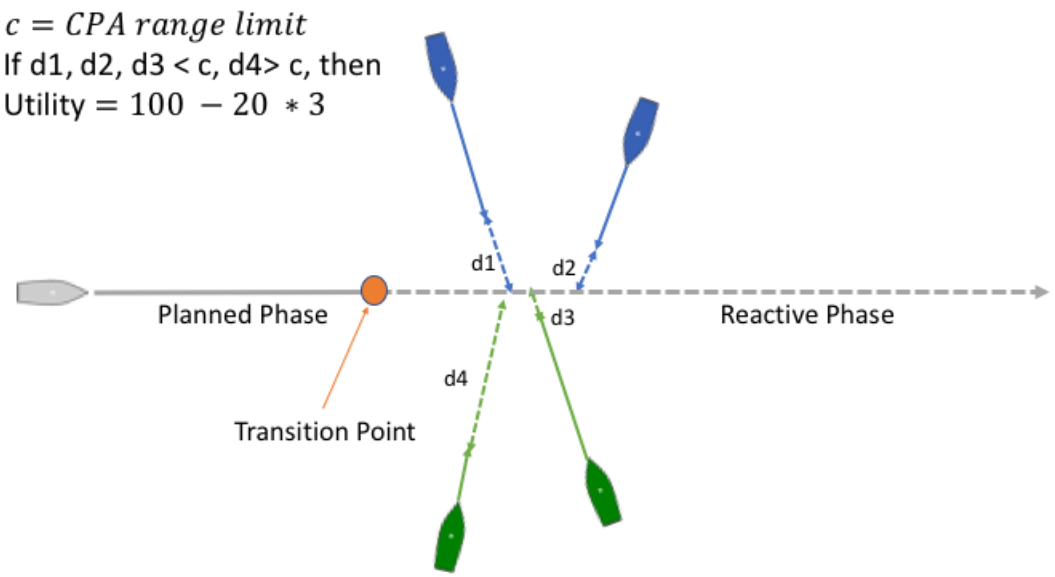


Figure 3-9: Utility calculation based on number of contacts in range.

For this research the naive utility calculation proves to be sufficient. This calculation can be appropriately scaled if the maximum density is greater than 4.

3.2.3 DensityCount Behavior

Like other MOOS behaviors, the DensityCount behavior is implemented as a C++ class with IvP Helm having one or more instances at run-time. The properties and implemented functions of a particular behavior are partly derived from the IvPBehavior superclass, which includes five virtual functions what are can be overloaded [6]:

- The *setParam()* function: parameter-value pairs are handled to configure a behavior's unique properties distinct from its superclass.
- The *onRunState()* function: the meat of a behavior implementation, performed when the behavior has met its conditions for running, with the output being an

objective function and a possibly empty set of variable-value pairs for posting to the MOOSDB.

- The *onIdleState()* function: what the behavior does when it has not met its run conditions. It may involve updating internal state history, generation of variable-value pairs for posting to the MOOSDB, or absolutely nothing at all.
- The *onIdleToRunState()* function: invoked once by the helm upon transitioning from the idle to running state.
- The *onRunToIdleState()* function: invoked once by the helm upon transitioning from the running to idle state.

For the DensityCount behavior, *setParam()* and *onRunState()* are overloaded to achieve desired autonomy function.

Configuration Parameters

The DensityCount behavior overloads the *setParam()* function with parameter values that are relevant to the planned phase, as shown in Figure 2-4. The full set of configuration parameters that gives DensityCount's desire autonomy features are listed below:

- Specific to DensityCount behavior
  - *polygon*: A list of vertices that forms a convex polygon that represents the high density area that the vehicle is approaching.
  - *transition_distance*: Distance in the direction to the goal from the vehicle to the closest boundary of the polygon.
  - *transition_period*: Desired time to be spent in the planned phase. The transition period multiplied by vehicle speed gives the distance from the transition to the activation point. The DensityCount behavior becomes active, i.e. produces an objective function, after reaching the activation point. It will become idle after reaching the transition point.

- *visual_hints*: A parameter that sets the appearance of objects published by the DensityCount behavior for use in the pMarineViewer application.

- Inherited from IvPBehavior superclass:

  - *name*: The name of the behavior - should be unique between all behaviors. This allows multiple instances of DensityCount behaviors to be run at the same time.

  - *priority*: The priority weight of the produced objective function. The default value is 100. Although possible, the DensityCount behavior is not implemented to determine its own priority weight. Instead, various experiments were run to determine an appropriate weight for this behavior.

  - *duration*: The time in seconds that the behavior will remain running before declaring completion. If no duration value is provided, the behavior will never time-out. The clock starts ticking once the behavior satisfies its run conditions (becoming non-idle) the first time. DensityCount is configured to never time-out, but only switch between running and idle states.

Runtime Processes

The DensityCount behavior overloads the *onRunState()* function to accomplish the primary work, to produce an objective function that favors speed choices that lead to lower maximum density. This objective function is an instance of the class IvPFunction, and a behavior generates an instance and returns a pointer to the object in the following function:

```
IvPFunction* onRunState()
```

This function is called automatically by the helm on the current iteration if the behavior is deemed to be in the running state. When called, if the return value is not a null pointer but a pointer to an IvPFunction, then the behavior is considered to be in the active state.

Within DensityCount behavior's *onRunState()* function several key functions are performed. First, the behavior receives information from the information buffer, the single source of outside information for all behaviors instantiated by IvP Helm. Individual behaviors sign up for what it is needed similar to the way MOOS applications register for variables in MOOSDB. Similar to the DensityCounter class, the DensityCount behavior reads the following variables from the *info_buffer*:

- NAV_X: current x coordinate of ownship
- NAV_Y: current y coordinate of ownship
- NAV_HEADING: current heading of ownship
- NAV_SPEED: current speed of ownship
- TRANSIT_SPEED: planned speed of ownship. Used to calculate total distance to be spent in the planned phase
- GOAL_POINT: location of the next waypoint.

Second, *onRunState()* implements a check on the relative position of ownship in relation to the activation point and transition point. It will further post visual confirmation of those two points as well as a vector going from ownship to the transition point. Also posted is the polygon representing the traffic zone. These visual objects are shown in Figure 3-10.

Third, if it is determined that ownship is in the planned phase, *onRunState()* produces an objective function using the ZAIC tools available in MOOS-IvP. Recall that an objective function is a piece-wise linearly defined function where each piece has an upper and lower boundary (or interval) on the decision space and linear function defined over the piece. The DensityCount behavior's objective function is defined over the speed domain and is bounded by ownship's capabilities. The speed domain is comprised of equally spaced discrete points, and therefore each piece is defined over a finite set of points in the domain. The ZAIC_Vector tool is used for generating IvP

Figure 3-10: Snapshot of vehicles and environment rendered by pMarineViewer. Nelson is ownship traveling west to east through a traffic lane. Nelson's goal point is a yellow point on the middle right side of this picture. The boundary of the traffic lane and the vector that points from Nelson to the transition are both produced by the DensityCount behavior

functions over one variable, in this case speed, given some number of explicit domain-range mappings. The information in DENSITY_UTIL is just such a mapping. Once parsed, DENSITY_UTIL becomes two equally sized vectors; a vector of speed and a vector of associated utilities. The ZAIC_Vector tool then creates an objective function that typically has a piece per given domain-range pair, where the slope of each piece simply approximates the domain-range characteristics for domain-range pairs not explicitly given. An example function created by ZAIC_Vector tool is shown in Figure 3-11:

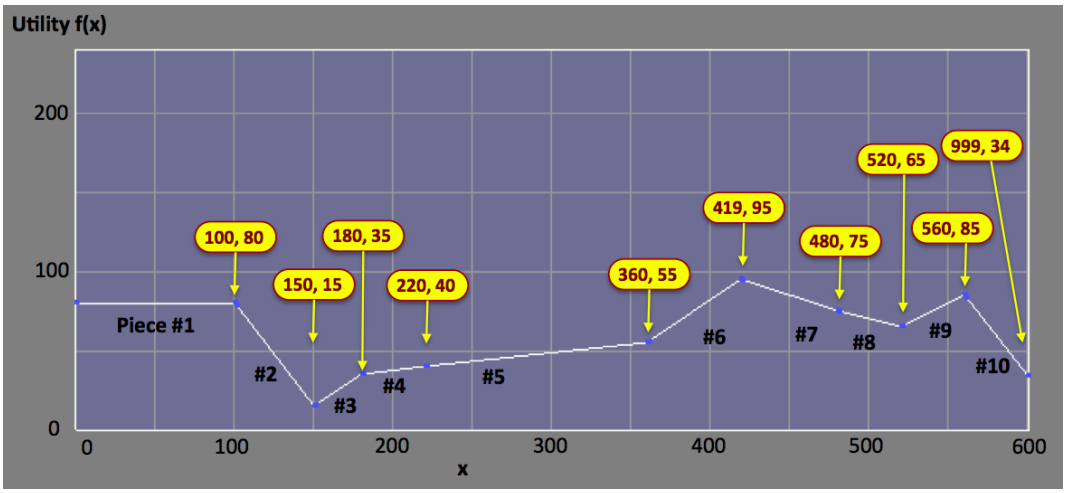


Figure 3-11: The ZAIC_Vector tool: defines an IvP function over one variable defined by a set of explicit domain-range mappings. Figure from [6].

Interaction with existing behaviors

The DensityCount behavior interacts with other behaviors in several capacities. At the very least, it must know where ownship is going next. While this information is obtained from the helm's *info_buffer*, it is originally supplied by the Waypoint behavior. The DensityCount behavior also needs to know the planned transit speed, which is typically supplied by the Waypoint behavior as well. The most important interaction, however, is not informational, but via the IvP solver. Figure 3-12 demonstrates the hierarchy of the Waypoint behavior, DensityCount behavior, and AvdColregs behavior during the various phases of navigation.

During the pre-planned phase, generally only the Waypoint behavior is active. This reflects a low contact density in the vicinity of ownship. If there are vessels that come within range and present a risk of collision, an AvdColregs behavior will be spawned and influence helm decisions. If correctly configured, during dire situations the AvdColregs behavior instances will dictate the helm decision. But such situations are rare during the pre-planned phase.

During the planned phase, the DensityCount behavior becomes active. It produces an objective function that penalizes speed choices that lead to higher maximum density inside the high vessel-density traffic zone. This objective function is designed

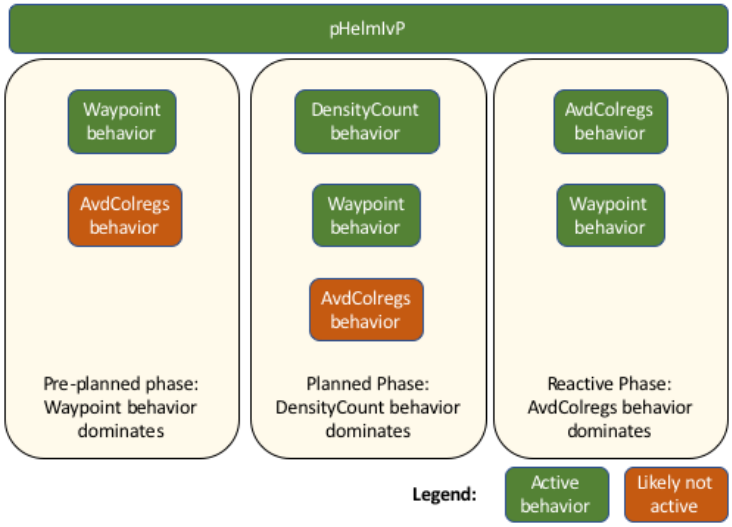


Figure 3-12: The relative dominance of each behavior during ownship's pre-planned phase, planned phase, and reactive phase.

to significantly influence IvP helm's final speed decision by setting DensityCount's priority weight greater than that of the Waypoint behavior during the planned phase. Furthermore, since DensityCount behavior's objective function is only over the speed domain, it will never force ownship to deviate from its planned course. Since similar danger exists that might require spawning an instance of the AvdColregs behavior, it is also imperative that the DensityCount behavior's priority weight is set no higher than the max priority weight of the former.

Finally during the reactive phase, since traffic density may be high, it is very likely that the AvdColregs behavior(s) will dominate the helm decision. However, the Waypoint behavior is still active, thereby guiding ownship towards the goal even during evasive maneuvers. Here DensityCount behavior's usefulness is voided. Quite the opposite, care is taken to configure the transition distance so that DensityCount behavior does not interfere with AvdColregs behavior in high density areas. The mission can also be configured so that when the AvdColregs behavior is active, it posts an update to MOOSDB that causes the DensityCount behavior to become idle.

Chapter 4

Experimental Setup and Results

Chapter 2 demonstrated that challenging real world navigation problems require a trait called forehandedness. The argument is made that Algorithms 1 and 2 combined can emulate forehandedness by taking a cautious approach to a high density area. In Chapter 3, these algorithms are implemented in MOOS-IvP through the pTrafficDensity application and the DensityCount behavior. This chapter will detail the process of setting up simulations that validate the effectiveness of Algorithms 1 and 2.

4.1 MOOS Mission Basics

A basic MOOS mission is launched by starting several MOOS processes that connect to a common MOOSDB forming a MOOS community. An example Alpha mission is shown in Figure 4-1. The MOOS applications in this community are configured with a mission (.moos) file comprising of configuration blocks for each of the applications. These configuration blocks differ by application, but at the minimum each application has the configuration parameters *CommsTick* and *AppTick*. The former configures how often the communications thread talks to the MOOSDB and the latter how often the main process inside the application, *Iterate()* function, will be called. The MOOS behaviors in this community are similarly configured with a behavior (.bhv) file. The main purpose of this file is to configure the IvP helm and associated behaviors with

essential information such as name, priority, and other behavior specific parameters. The behavior file can further structure the autonomy mission into modes of operation for more efficient management of behaviors.

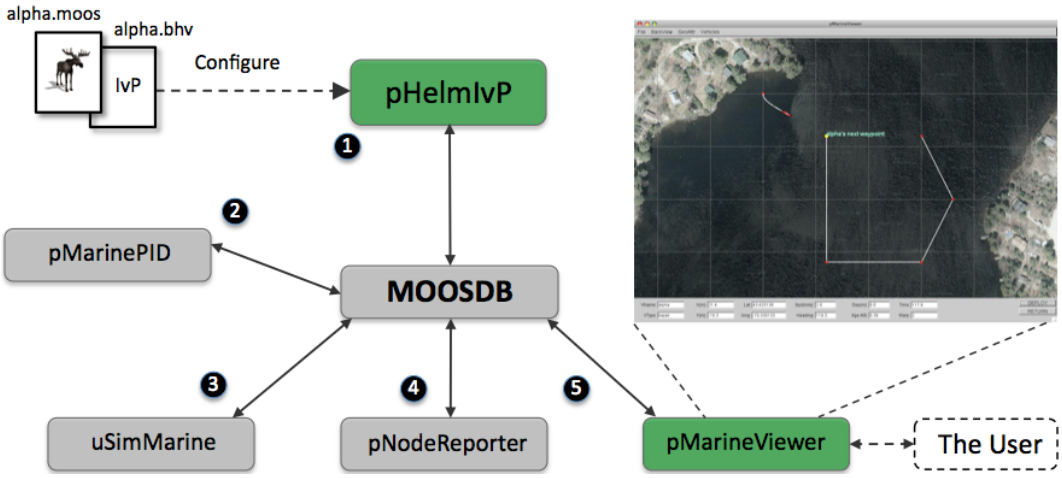


Figure 4-1: The MOOS processes in the example "alpha" mission: In (1) The helm produces a desired heading and speed. In (2) the PID controller subscribes for the desired heading and speed and publishes actuation values. In (3) the simulator grabs the actuator values and the current vehicle pose and publishes a set of MOOS variables representing the new vehicle pose. In (4) all navigation output is wrapped into a single node-report string to be consumed by the helm and the GUI viewer. In (5) the pMarineViewer grabs the node-report and renders a new vehicle position. The user can interact with the viewer to write limited command and control variables to the MOOSDB. Figure from [6].

This research simulates multi-vehicle missions, consisting of multiple MOOS communities each configured similar to the Alpha mission shown in Figure 4-1. Like the Alpha mission, each vehicle has its own MOOSDB that interacts with notable applications like pMarinePID, uSimMarine, pNodeReport. pMarinePID is a PID controller that converts high-level control decisions from the helm into low-level actuator commands. The uSimMarine application is a simple 3D vehicle simulator that updates vehicle state, position and trajectory, based on the present actuator values and prior vehicle state. uSimMarine can also accept external drifts that are adjusted dynamically by other MOOS applications based on any criteria wished by the user and developer. No environmental drifts are simulated in this research.

Multi-vehicle missions differ from the Alpha mission in the topology of MOOS

communities. Figure 4-2 shows how the missions in this research are set up. Each node manages its own MOOS community. In all missions, contacts 1-4 run the same applications and behaviors, differing only in configurations such as vehicle names, transit speeds, starting positions etc. In addition to those applications and behaviors, ownship also runs the pTrafficDensity application and the DensityCount behavior. The Shoreside MOOS community acts as command and control which runs the only instance of pMarineViewer, a powerful tool that is discussed in detail below.

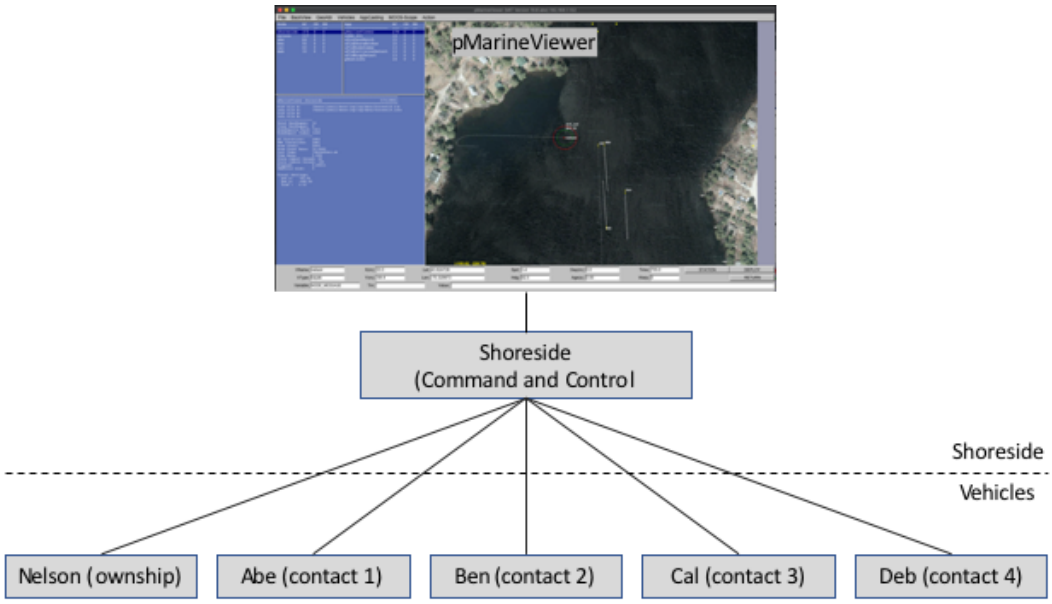


Figure 4-2: Shoreside to multi-vehicle topology: Five vehicles are deployed with each vehicle maintaining connectivity to a shoreside command and control computer. Each node (vehicles and the shoreside) are comprised of a dedicated MOOS community. pMarineViewer is only run on the shoreside for mission monitoring.

4.2 Tools Used

4.2.1 pMarineViewer

The primary application within the MOOS-IvP suite that allows users to monitor simulated and in-water mission testing is the pMarineViewer application. It is a MOOS application written with FLTK and OpenGL for rendering vehicles and associated information and history during operation or simulation [6]. Users can also

manipulate a geo display, to see multiple vehicle tracks and monitor information on selected vehicles, shown in the right hand side of Figure 4-3. Information that can be observed in real-time include vehicle positions, tracks, status, and which behaviors are active on a selected vehicle.

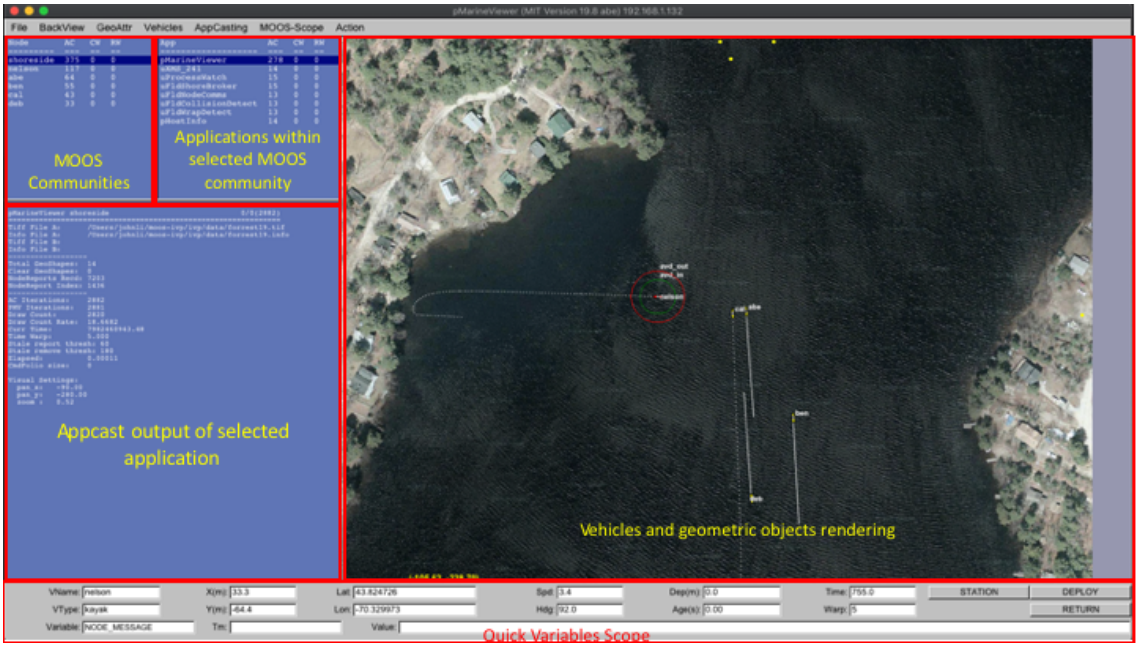


Figure 4-3: Basic Layout of pMarineViewer application.

Both vehicles and the shoreside station MOOS community can be selected on upper left box of pMarineViewer, shown in Figure 4-3. Once a MOOS community is selected the list of Appcasting capable MOOS applications in that community appears in the box to the immediate right. Below the community and application selection boxes are the Appcast output of the selected application.

At the bottom right of Figure 4-3, there are three action buttons called DEPLOY, STATION, and RETURN. These configurable buttons give the user overall control for each mission and served as the first line of safety for mission execution. All vehicles received orders to deploy, return, station keep, or come to all-stop through operator action in pMarineViewer. Other configurable functions includes: (a) pull-down menu actions, (b) contextual mouse poking with embedded OPAREA information, and (c) commander pop-up window [6].

While the pMarineViewer application is invaluable as the primary real-time mon-

itoring tool, it does not keep a record of events. The particular events that this research is interested in are close encounters between vehicles. The uFldCollisionDetect application offers this capability.

4.2.2 uFldCollisionDetect

The uFldCollisionDetect application is run on the shoreside and monitors pairs of vehicles for encounters that come within a certain range. The closest point of approach (CPA) is noted when the range between two vehicles transitions from closing to opening. Depending on the CPA value, one of three events may be declared, either an encounter, a near miss, or a collision, depending on user configured range parameters [6].

A note on CPA. In MOOS-IvP, CPA is calculated using the built-in CPA engine, which conducts rapid CPA calculation assuming that both vessels maintain their current course and speed. Since CPA is not only used by uFldCollisionDetect but by a wide variety of MOOS behaviors, the CPA engine has been optimized for efficiency by smart caching of current vehicle positions and the current position and trajectory of other vehicles [9].

For missions in this research, all three key range threshold parameters, encounter_range, near_miss_range, and collision_range, are set. Encounter_range is the CPA range beyond which two vehicles are considered to be too far away to be regarded as having had an encounter. Outside this range it is a non-event.

The second parameter, near_miss_range, determines a CPA range within which an encounter is considered to be a near miss. Encounters even closer, with the range specified by collision_range, are categorized as collisions [6].

The default values are (in meters):

```
encounter_range = 20
near_miss_range = 6
collision_range = 3
```

Subjected to:

$$encounter\_range \geq near\_miss\_range \geq collision\_range$$

For the various experiments ran for this research, these three ranges are adjusted to capture encounters at different distances. Upon each encounter less than or equal to the encounter_range, an encounter counter is internally incremented. Likewise for near misses and collisions. A collision encounter will not however also increment the near miss counter. These counters are maintained globally for all vehicles. The important roles these ranges play in analyzing the performance of the DensityCount behavior will be discussed in later sections.

4.2.3 alogcd

The alogcd utility is a command line post-mission analysis tool that complements the capabilities of uFldCollisionDetect. This tool scans log files generated by MOOS missions and tallying the number of encounters, near misses, and collisions as set by uFldCollisionDetect. These tallies comprise the main source of data that forms the results of this research. An example output file is shown below:

```
Analyzing collision encounters in file : LOG_SHORESIDE_21_7_2020_____
14_00_01/LOG_SHORESIDE_21_7_2020_____14_00_01.alog


+++++++++++++++++++++++++ (250,000) lines
+++++++++++++++++++++++++ (500,000) lines
+++++++++++++++++++++++++ (750,000) lines
+++++++++++++++++++++++++ (1,000,000) lines
+++++++++++++++++++++++++ (1,250,000) lines
+++++++++++++++++++++++++ (1,500,000) lines
+++++++++++++++++++++++++ (1,750,000) lines
+++++++++++++++++++++++++ (2,000,000) lines
+++++++++++++++++++++++++ (2,250,000) lines
++++++++
2,332,216 total alog file lines.
```

```
=======================================
Collision Report:
=======================================
Encounters:  102  (avg 29.43 m)
Near Misses: 10  (avg 0.00 m)
Collisions:  6  (avg 1.99 m)
Collision Worst: 0.60
```

4.2.4 alogview

The final tool discussed here is the alogview application. Similar to pMarineViewer, alogview allows the users to manipulate a geo display, but it does so post-mission. The power of alogview is that the full state is maintained across all vehicles for both playing back sequentially and rewinding or jumping to any arbitrary point in time. The alogview tool also gives users the ability to view any time-series data that was recorded during the mission. Furthermore, users can use the IPFPlot window in alogview to render objective functions produced by individual behaviors throughout the mission by vehicle. This function is frequently used in this research and snapshots of the IPFPlot window will be featured in the results analysis.

4.3 Baseline Mission

The baseline mission is the first set of missions designed to validate the utility of the planned phase. To reconstruct a situation similar to crossing the Strait of Mallaca shown in Figure 2-2, ownship Nelson is tasked with crossing a high density traffic lane from west to east. Per assumption 1 of Section 2.2.1, dense traffic is enclosed in the traffic lane bounded by a polygon. Inside the traffic lane, contacts Abe, Ben, Cal, and Deb traverse north and south repeatedly. In this experiment, all vehicles traverse to their respective destinations at a random speed and without any evasive maneuvering. The experiment is set up so that the experimental group runs the pTrafficDensity application and the DensityCount behavior while the control group

does not run the DensityCount behavior (pTrafficDensity still runs but does not influence the behavior of Nelson in the control runs). A snapshot of a control group run is shown below in Figure 4-4.

Figure 4-4: Snapshot of a control group run. All vessels follow pre-planned waypoints and and speed. No collision avoidance is conducted.

In the experimental group, the DensityCount behavior becomes active during the planned phase and either slows or speeds up Nelson until the transition point. As discussed in Section 3.2.3, DensityCount publishes several geometric artifacts for visual feedback. This results in a mission that is immediately identifiable as shown in Figure 4-5

Figure 4-5: Snapshot of a experimental group run. All vehicles follow pre-planned waypoints and and speed. No collision avoidance is conducted.

4.3.1 Batch Runs # 1

Configuration

For this batch of experiments, each run is 5 hours long. At the start Nelson goes from west to east through the traffic lane at a randomized transit speed between 1 meters/second and 5 meters/second. The start point and end point for Nelson is fixed (-100,-75 to 270, -75). Vehicles Abe and Ben both start from random points in 20 meters x 20 meters boxes in the northern end of the traffic lane. Like Nelson, Abe and Ben's transit speed are also randomized between 1 meters/second and 5

meters/second. Vehicle Cal and Deb start from random locations in the south with random speed. This setup presents a variety of navigational situations while keeping the general flow of traffic inside the traffic lane.

Upon reaching their destinations, each vehicle will turn back around. Since the four contact vehicles have random transit speeds, on Nelson's way back, he is presented with another traffic lane crossing having very likely a new and unique navigational situation. With each crossing, Nelson will face a maximum of 4 vehicles at the same time and minimum of 0. At an average speed of 3 meters/second, Nelson can go from the starting point to the end point in approximately $370m \div 3m/s \approx 123sec \approx 2mins$. This means for the experimental run time of 5 hours, Nelson will make about 146 one-way trips with a "maximum" of 584 encounters, i.e., if Nelson encounters all four contacts at every crossing.

Reaching the "maximum" is highly unlikely when the encounter threshold is 50 meters. The reason is that the traffic lane is 300 meters long, so it is very likely one or more vessels can be traveling away from Nelson when he enters the traffic zone and never cause an encounter.

The reason 584 is not exactly the theoretical maximum number of encounters is due to some edge cases. Suppose a contact vehicle is assigned a speed close of 5 meters/second while Nelson is assigned 1 meters/second, it is possible for that contact to cause 2 encounters during Nelson's crossing. This scenario is unlikely because the contact vehicle would have to traverse through Nelson's track to the upper or lower edge of the traffic lane then come back down. This would take approximately $(300m - 50m) \div 5m/s = 50sec$, while it also takes Nelson $50m \div 1m/s = 50sec$ to cross the traffic lane with a width of 50 meters.

In this batch, the collision avoidance behavior is inactive. The first 6 runs are set as control, where the DensityCount behavior is not on. They are runs 1 through 6 in Table 4.1. When the DensityCount behavior is not active, the Range Limit column will have a corresponding zero entry. Runs 7 through 16 have the DensityCount behavior enabled, with the Range Limit c set at 50 meters. Runs 17 through 20 are DensityCount enabled as well, with the Range Limit c set at 70 meters. Full run

results are shown below in Table 4.1.

Table 4.1: Batch # 1 Raw Results

Run #	Encounters (50 m)	Near Misses (10 m)	Collisions (5 m)	Range Limit (in m)
1	343	18	17	0
2	372	36	17	0
3	321	22	17	0
4	366	38	18	0
5	311	27	17	0
6	357	44	14	0
7	244	13	15	50
8	315	31	22	50
9	227	10	13	50
10	314	30	14	50
15	241	22	14	50
16	317	40	15	50
17	380	40	15	70
18	227	18	13	70
19	304	36	16	70
20	279	17	9	70

The eye test says that the two experimental groups are doing better than the control runs (1-6). Further analysis is done by averaging the number of encounters, near misses, and collisions for each group and compared to the control. This result is shown below in Table 4.2:

Table 4.2: Batch # 1 Analysis

Group	Encounters (50 m)	*Change* %	Near Misses (10 m)	*Change* %	Collisions (5 m)	*Change* %
Control	345		30.8		16.7	
c = 50 m	276.3	−19.9	24.3	−21.1	15.5	−7
c = 70 m	297.5	−13.8	27.8	−10	13.3	−20.5

When Range Limit c is set at 50 meters, the DensityCount behavior reduced the number of encounters by nearly 20%, number of near misses by 21% and number

of collision by 7%. Range Limit c is set at 70 meters, the DensityCount behavior reduced the number of encounters by nearly 14%, number of near misses by 10% and number of collision by 20%.

This result is promising enough to warrant further testing but not enough to prove performance. The confounding variable is Nelson's transit speed. Recall that Nelson's speed is randomly selected between 1 meters/second and 5 meters/second, and that transit speed largely dictates how many trips across the traffic lane Nelson will make. With only 6 long runs per group, even though the type of encounters are randomized, the chance for encounter could be skewed by the speed assigned to Nelson in these trials. Batch runs #2 seek to reduce this problem.

4.3.2 Batch Runs # 2

This batch of runs has nearly identical settings as Batch # 1. The only adjustment made is that the boundary of the traffic lane has been expanded 30 meters north and south so the length is now 360 meters. Run time is still 5 hours. All vehicles have randomized speed from 1 meters/second to 5 meters/second. Twenty runs each for the control and experimental groups were completed. For the experimental group Range Limit c is set at 50 meters. Raw results are shown in Table 4.3, and similar analysis is offered in Table 4.4.

While some improvement is shown in encounter and collision numbers, there has been no improvement for near misses. Post mission analysis of these runs reveals two factors that affected the performance of DensityCount:

1. **Contact Position Awareness**: Communications range between vehicles is set at 120 meters in these simulations. This effectively limits Nelson's contact detection range to 120 meters. To effectively calculate future density, Nelson needs to pick up these contacts at a higher range.

2. **Priority Weight**: Priority weight for DensityCount is set at 150, versus the default priority weight of 100 for the Waypoint behavior. This gives DensityCount relatively higher weight, but will not quite dominate the planned phase.

Table 4.3: Batch # 2 Raw Results

Run #	Encounters (50 m)	Near Misses (10 m)	Collisions (5 m)	Behavior Status (in m)
1	208	19	12	0
2	203	10	15	0
3	215	14	15	0
4	222	13	11	0
5	311	27	17	0
6	219	18	12	0
7	239	18	11	0
8	244	19	8	0
9	213	17	15	0
10	212	17	14	0
11	199	12	16	0
12	257	24	20	0
13	226	10	15	0
14	182	17	7	0
15	220	16	11	0
16	224	20	16	0
17	241	13	9	0
18	199	15	17	0
19	240	16	15	0
20	328	34	21	0
21	172	14	9	70
22	189	22	9	70
23	162	11	12	70
24	262	18	14	70
25	181	25	14	70
26	169	16	6	70
27	219	17	18	70
28	235	14	9	70
29	173	16	6	70
30	207	23	10	70
31	225	27	17	70
32	200	10	10	70
33	220	18	21	70
34	216	22	9	70
35	199	16	14	70
36	199	14	12	70
37	214	11	8	70
38	222	16	14	70
39	253	22	12	70
40	196	17	7	70

Table 4.4: Batch # 2 Analysis

Group	Encounters (50 m)	*Change* %	Near Misses (10 m)	*Change* %	Collisions (5 m)	*Change* %
Control	230.1		17.45		13.85	
c = 50 m	205.65	−10.6	17.45	0	15.5	−16.6

4.3.3 Batch Runs # 3

For this batch of experiments, the communications range is expanded to 500 meters. This change in setting effectively gives Nelson awareness of contact locations anywhere within the operations area. Further more, the priority weight for DensityCount is bumped up to 250, giving it more influence over Nelson's helm decision and ultimate the speed ordered. For control, Batch # 2's control runs will suffice because the communications range and priority weight does not affect how vehicles operate in those runs. The raw data for a group of 20 runs is shown below in Figure 4.5.

Using the same analysis as Batches #1 & #2, Figure 4.6 summarizes the impact of running the DensityCount behavior. With near perfect knowledge of the contact locations and a dominating priority weight, DensityCount reduced ownship's encounters by 17.2%, near misses by 26.6%, and collisions by 27.1%.

4.3.4 Summary of Results

The baseline mission set out to prove the effectiveness of the DensityCount behavior in the absence of any collision avoidance maneuvers. Batch # 3's results, shown in Figure 4.6, indicate that DensityCount behavior produced noticeable improvements over the control group in all three categories of encounters measured.

Exactly what has been improved upon is worth exploring. If used as a collision avoidance algorithm, DensityCount's performance is less than stellar. In run #1 of Batch # 3, shown in Table 4.5, Nelson effectively collided with another vessel 17 times out of about 150 traversals (the log shows Nelson's transit speed is 3 meters/second) through the traffic zone. However, on average DensityCount did reduce the number

Table 4.5: Batch # 3 Raw Results

Run #	Encounters (50 m)	Near Misses (10 m)	Collisions (5 m)	Behavior Status (in m)
1	234	18	17	50
2	102	10	6	50
3	226	13	15	50
4	82	6	1	50
5	197	10	10	50
6	73	10	4	50
7	158	12	13	50
8	106	4	6	50
9	188	10	9	50
10	271	21	12	50
11	224	9	13	50
12	257	22	17	50
13	152	12	10	50
14	234	20	15	50
15	172	9	4	50
16	276	17	12	50
17	175	8	5	50
18	263	19	14	50
19	214	4	4	50
20	206	21	15	50

of collisions from the control group.

One way to reconcile these two facts is to consider the number of encounters, near misses, and collisions as a proxy measure for the *risk* of collision. After all, at the most basic level, to determine how risky a situation is, one must ask how bad the outcome would be if no action is taken? Going back to the navigational picture presented in Figure 2-2, a good reason to assess that vessel E (anchored) and vessel D (transiting inside the TSS) are in a low risk situation is that *no action* is required of them.

The baseline mission simulates exactly that. Since the data are collected with neither Nelson nor the contacts making any attempt to avoid collision inside the traffic zone, a higher number is a direct measure of higher collision risk. Introducing the planned phase and DensityCount prior to the traffic zone reduces the number

Table 4.6: Batch # 3 Analysis

Group	Encounters (50 m)	*Change* %	Near Misses (10 m)	*Change* %	Collisions (5 m)	*Change* %
Control	230.1		17.45		13.85	
c = 50 m	190.5	−17.2	12.8	−26.6	10.1	−27.1

encounters of all kinds, and therefore reduces the *risk* of collision.

4.4 COLREGS Mission Configuration

The success of the baseline mission indicates that the DensityCount behavior can reduce the risk of collision, provided that the number of encounters of all kinds is a good proxy measure for that risk. The COLREGS missions seek to validate the performance of the DensityCount behavior in a different way, by testing to see whether inserting a planned phase between path following and collision avoidance improves the overall safety of the vehicles involved.

In this experiment, vehicles conduct collision avoidance only if ownship is involved. This simulation choice is made to keep the contact vehicles on relatively stable north and south tracks. Another decision made is that the AvdColregs behavior has the same setting on all vehicles. This level of reciprocity is not always the case in the real world, as different mariners might have different risk perceptions and could take actions earlier or later.

Besides the introduction of collision avoidance, the setting of the COLREGS missions versus the baseline missions are largely the same. Wherever a particular parameter is different, it will be pointed out and explained in detail in the batch runs belonging to that mission.

4.4.1 Batch Runs # 4

Configurations

With multiple behaviors interacting with each other, the mission planner must keep a close eye on the configuration parameters governing each. Here the focus is on the AvdColregs and DensityCount behaviors as well as the pTrafficDensity application. Important configuration parameters for this batch are shown below:

```
For AvdColregs behavior:    For DensityCount behavior:
pwt               = 300     pwt                 = 200
completed_dist    = 45      transition_distance = 35
max_util_cpa_dist = 20      transition_period   = 60
min_util_cpa_dist = 10
pwt_inner_dist    = 10      For pTrafficDensity application:
pwt_outer_dist    = 35      range_limit         = 45
```

These parameters and their functions were first discussed in isolation in Chapter 3. When put together, the above settings have several interesting features. First, with the range_limit set at 45 meters, but the encounter distance set at 50 meters, the density calculation is less cautious than the events we are measuring. Second, the priority weight of the DensityCount behavior is twice as much as the default Waypoint behavior's weight of 100, but much smaller than the maximum weight of the AvdColregs behavior. Third, the transition_distance is set at 35 meters while the pwt_outer_dist is also 35 meters. This means that notionally the DensityCount behavior will not interfere with AvdColregs provided that no contacts veer out of the traffic lane.

The raw results are broken into two tables. Table 4.7 shows 25 runs without DensityCount and Table 4.7 shows 25 runs with DensityCount. Each of the runs is 2.5 hours long, allowing on average 73 one-way transits through the traffic lane and a maximum of 284 encounters. Analysis is shown in Table 4.9.

As shown in Table 4.9, there is a significant reduction in the number of encounters. Not only are the encounters less frequent, but when an encounter does occur, the CPA

Table 4.7: Batch # 4 Raw Results, without DensityCount

Run #	Encounters (50 m)	Average Encounter CPA (in m)	Near Misses (10 m)	Collisions (5 m)
1	158	32.49	0	0
2	139	31.59	0	0
3	138	28.79	2	1
4	104	31.56	0	0
5	170	29.9	3	1
6	125	31.93	0	0
7	157	31.07	1	0
8	138	25.92	0	0
9	183	29.16	0	0
10	118	29.86	0	0
11	155	30.26	0	0
12	151	28.38	2	0
13	129	29.81	0	0
14	111	29.96	0	0
15	149	29.09	1	0
16	132	30.59	0	0
17	133	29.24	0	0
18	160	29.37	0	0
19	170	29.19	0	0
20	129	28.44	0	0
21	155	27.58	2	0
22	190	30.83	0	0
23	131	29.12	0	0
24	180	30.28	0	0
25	120	28.98	0	0

of that encounter tends to be higher. This is indicated by the 11.1% increase in the CPA distance in column 4 of Table 4.9. The qualitative improvements in addition to the quantitative improvements in encounters is strong evidence that implementing the planned phase can help the vessel in finding sea room.

However, the number of near misses jumped up by more than twofold. This suggests that either the DensityCount behavior is putting Nelson into harm's way more often inside the traffic zone or something else is at work. Various alogview playbacks suggests that the problem lies in the configuration of the AvdColregs behavior. An

Table 4.8: Batch # 4 Raw Results, with DensityCount and $c = 45$ m

Run #	Encounters (50 m)	Average Encounter CPA (in m)	Near Misses (10 m)	Collisions (5 m)
1	58	32.63	0	0
2	131	30.78	2	0
3	62	34.12	2	0
4	123	34.23	1	0
5	57	35.13	0	0
6	152	33.4	1	0
7	105	31.6	1	0
8	52	40.91	0	0
9	112	33.29	1	0
10	56	34.48	0	0
11	137	35.76	1	0
12	76	29.75	1	0
13	148	32.94	10	2
14	68	32.11	0	0
15	139	32.48	3	0
16	59	33.64	0	0
17	155	33.66	2	0
18	38	35.65	0	0
19	78	35.98	0	0
20	115	32.22	3	0
21	125	32.33	1	0
22	45	31.85	0	0
23	176	30.59	0	0
24	46	34.24	0	0
25	173	31.51	0	0

Table 4.9: Batch # 4 Analysis

Group	Encounters (50 m)	*Change* %	Average Encounter CPA (in m)	*Change* %	Near Misses (10 m)	*Change* %	Collisions (5 m)	*Change* %
Control	139.4		29.7		0.4		0.1	
c = 45 m	99.4	−28.7	33	11.1	1.2	124.2	0.1	0

example playback moment is shown in Figure 4-6.

Manual playback of the least successful runs, i.e. runs #13, #15, and #20 reveals that the interference between AvdColregs behavior and DensityCount is only a minor

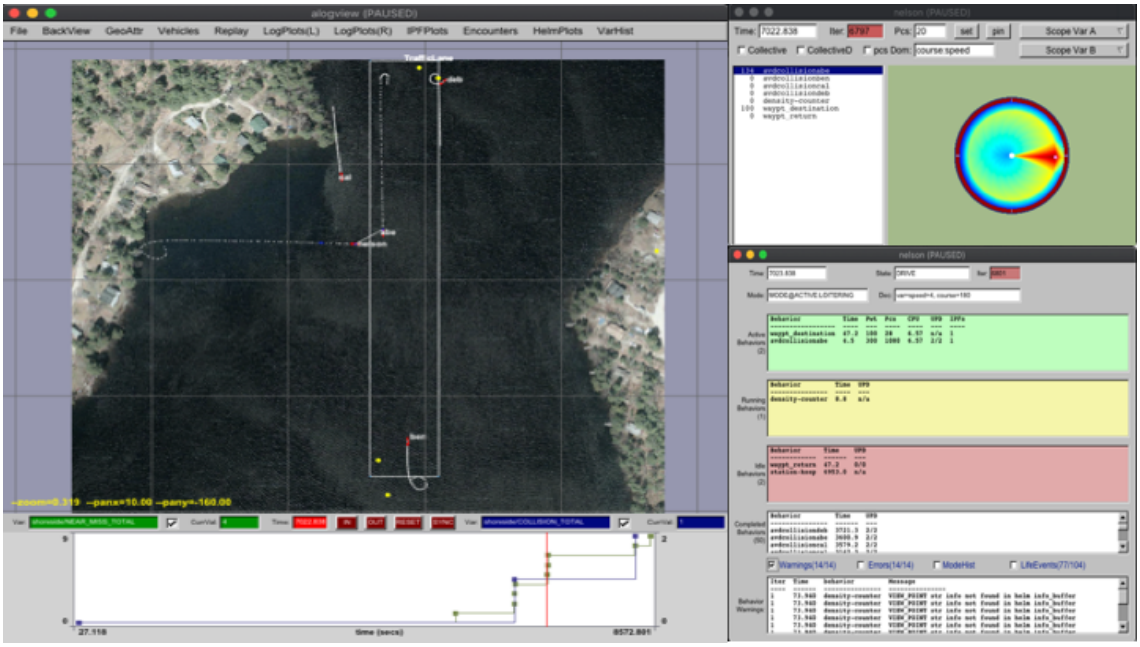


Figure 4-6: Alogview post mission analysis. Left window: main alogview shows the geo display. Upper right window: IPFPlots showing behavior objective functions and associated priority weight. Lower right window: HelmPlots showing detailed behavior information.

contributor to the increase in near misses, accounting for just 2. The near misses in run #13, by far the worse run, appear to be caused by the relatively high speeds Nelson and Abe are assigned: both 4 meters/second. This reveals that for a maximum speed of 5 meters/second, it is not sufficient to have a pwt_outer_dist set to 35 meters and max_util_cpa_dist set to 30 meters. This configuration means AvdColregs does not become active until vehicles are within 35 meters of each other, and when active, the AvdColregs behavior's influence over the helm decision does not become the highest until 30 meters. Under these conditions, vehicles have a difficult time avoiding near misses with a high waypoint transit speed.

4.4.2 Batch Runs # 5

This batch of experiments take the lessons learned from Batch #4 and configured the behaviors accordingly:

```
For AvdColregs behavior:     For DensityCount behavior:
pwt                = 300     pwt                 = 200
```

```
completed_dist    = 70      transition_distance = 70
max_util_cpa_dist = 30      transition_period   = 120
min_util_cpa_dist = 10
pwt_inner_dist    = 20      For pTrafficDensity application:
pwt_outer_dist    = 60      range_limit         = 70
```

Under this new configuration, AvdColregs behavior becomes active at 60 meters and completes at 70 meters, therefore alleviating the previous issue of high near misses. Transistion_distance for DensityCount has been bumped up to 70 meters to ensure that it does not interfere with AvdColregs and transition_period has been bumped up to ensure DensityCount has sufficient time to influence helm decisions.

In uFldCollisionDetect, the encounter_range has been increased to 70 meters and the near_miss_range has been increased to 15 meters to capture more interactions and reduce the chance of statistical outliers. pTrafficDensity's range_limit has also been increased accordingly to 70 meters.

A total of 84 runs were conducted for this batch. Raw data from these runs can be found in Table B.1 and Table B.2. The analysis of this data is shown below in Table 4.10

Table 4.10: Batch # 5 Analysis

Group	Encounters (70 m)	*Change* %	Average Encounter CPA (in m)	*Change* %	Near Misses (15 m)	*Change* %	Average Near Misses CPA (in m)	*Change* %
Control	208		43.8		0.3		12.7	
c = 70 m	170.8	−17.9	47.7	9	0.2	−34.9	11.6	−8.8

These results show notable improvements in almost all categories measured. The average number of encounters went down from 208 to 170.8, a 17.9% decrease. While the average encounter distance saw an increase of 9% from 43.8 meters to 47.7 meters. Collisions have been completely eliminated from both groups, a testament to the effectiveness of modifications made to the AvdColregs configuration parameters. Some near misses remain but they are relatively infrequent for both groups. The experimental group shows a reduction of almost 35% from the control group in the

number of near misses, but this is tempered by the fact that the average CPA in near misses went down by about 8.8%.

Post mission alogview playback of experimental runs with near misses reveals an interesting factor. The average transit speed for these runs (see runs # 3, 5, 7, 31, 35, and 38 in Table B.2) is 4.22 meters/second, a number much higher than the expected average value of 3 meters/second. With a speed of 4.22 meters/second, it will take Nelson only $(260m - 70m) \div 4.22m/s \approx 45sec$ to go from the start point to the transition point. This means even though notionally Nelson can spend 120 seconds in the planned phase, at higher speed that opportunity is not afforded. With only 45 seconds, the DensityCount behavior's influence over the time of arrival at the transition point is significantly reduced, preventing Nelson from arriving at an advantageous time.

4.4.3 Batch Runs # 6

This batch of runs seeks to give Nelson the opportunity to spend a longer time in the planned phase. The western and eastern waypoints for Nelson have both been moved 50 meters further from the traffic lane, thereby increasing the total distance from one waypoint to the boundary of the traffic lane to 310 meters. For the planned phase, the distance is the smaller of $310m - 70m = 240m$ or $transit\_speed \times 120$. At 240 meters distance, it represents a 26% increase over Batch #5.

```
For AvdColregs behavior:    For DensityCount behavior:
pwt               = 300     pwt                 = 200
completed_dist    = 70      transition_distance = 70
max_util_cpa_dist = 30      transition_period   = 120
min_util_cpa_dist = 10
pwt_inner_dist    = 20      For pTrafficDensity application:
pwt_outer_dist    = 60      range_limit         = 70
```

Holding all other configuration parameters the same as Batch # 5 (shown above) and changing only Nelson's waypoints, a total of 90 runs were conducted. Each run

is approximately 3 hours long, allowing on average 49 one-way trip across the traffic lane. This amounts to 196 encounters on average, if every crossing were met with all 4 contacts. Raw results from these runs can be found in Table B.3 and Table B.4. Analysis of the data is shown in Table 4.11 below:

Table 4.11: Batch # 6 Analysis

Group	Encounters (70 m)	*Change* %	Average Encounter CPA (in m)	*Change* %	Near Misses (15 m)	*Change* %	Average Near Misses CPA (in m)	*Change* %
Control	160.2		43.8		0.2		13	
c = 70 m	120.2	−25	49	12.9	0	−100	0	

Allowing Nelson to have a longer planned phase yields very positive results. The number of encounters went from 160.2 to 120.2, a 25% reduction. At the same time the average encounter CPA increased by 12.9%. This might not sound like much but the actual opening in average CPA is 5.2 meters, just slightly larger what is considered a collision in this environment. Furthermore, the experimental runs completely eliminated near-misses, which really showcases the power of the planned phase.

4.4.4 Summary of Results

The COLREGS missions set out to not only reduce the *risk* of collision, but also actual incidents of collisions and near misses. Through repeated testing and tuning of configuration parameters, various relationships between configuration parameters across different behaviors are revealed. First, the decision of when to activate the AvdColregs behavior, should largely depend on expected maximum relative closing speeds. In this experiment, setting pwt_outer_dist at 60 meters seems to be sufficient to handle vehicle speed of 5 meters/second or less.

Second, the relationship pwt_outer_dist and transition_distance plays a role in the number and severity of near misses. To set transition_distance equal to pwt_outer_dist would require traffic to stay inside designated traffic lane. Depending on how likely that is, an appropriate buffer may be added to transition_distance.

Alternatively, users might elect to turn off DensityCount behavior whenever the AvdColregs behavior is active.

Third, setting the transition_period parameter does not automatically guarantee that ownship will have the opportunity to use that time. The navigational constraint largely dictates how much time the vehicle can spend in the planned phase.

4.5 Additional Discussions

Two aspects of this implementation were not studied explicitly but are discussed below:

1. Effect on the Global Path: The DensityCount behavior's impact on the global path is positively correlated to its priority weight. Ultimately the IvP solver calculates what the optimal speed and course is for a particular iteration, weighing objective functions from all active behaviors. In practice, the DensityCount behaviors can cause the vehicle to speed up, in order to "shoot the gap", or slow the vehicle down to wait for a gap to appear. In this research, exact scenarios were not replayed with and without the behavior, rather an overall statistical summary of a variety of scenarios are presented in the results.

   It is possible to devise experiments that compare time required to achieve next waypoint or the total path lengths, but such experiments make decoupling the performance of the collision avoidance algorithm with that of the DensityCount behavior difficult. Part of what DensityCount enables the vehicle to do is avoid having to take evasive actions, actions which by definition cause the vehicle to deviate from planned course and cause inefficiencies. The mission planner must weigh the relative benefits of staying on planned waypoint time versus reducing the probability of evasive maneuvers.

2. Computational Cost: The DensityCount behavior is very light weight, since all it has to do is construct an objective function with the speed-utility vector produced by pTrafficDensity. pTrafficDensity runs in parallel with other MOOS

applications, and is configured to run its main process twice a second. In the experiments, pTrafficDensity consistently produces required outputs every iteration.

The functionality of pTrafficDensity can be expanded to evaluate a wider range of maneuvering options. This decision is left to the mission planner. It is probable that with an expanded decision domain, Algorithm 2 can take too long. This problem can be mitigated with configuration parameters: 1) Reduce the *AppTick* of pTrafficDensity, thereby allow it more time to complete its processes; 2) Increase the simulation time step, which proportionally reduces number of computations required for each simulation.

Chapter 5

Conclusions and Future Works

5.1 Conclusions

This thesis introduces the naval concept called "forehandedness", a loose synonym to foresight and prudence in matters pertaining to navigation and seamanship. A new approach seeking to emulate this concept call planned phased is described and then implemented using MOOS-IvP. Testing shows that by implementing the planned phase, *risk* of collision is reduced. Furthermore, when complemented by a well configured collision avoidance algorithm, the algorithm proposed can help to eliminate collisions and near misses.

The findings of this thesis are subjected to various limitations, which are listed explicitly in assumptions. While most of the assumptions are commonplace among collision avoidance literature, such as knowledge of contacts, predictable ship trajectories, etc, one assumption is unique to this research: the requirement that of all the areas a vessel traverses through, there are distinct pockets of high density and low density areas. This is partially inspired by the practice of navy ship manning extra watches prior to entering areas of known high traffic density or greater navigational hazards. USVs can also adopt a different "posture" prior to entering these areas.

A forward looking risk analyzer that evaluates maximum traffic density on a given track (speed and heading) is shown to be an effective tool in collision prevention. This effectiveness vindicates density calculation as unsophisticated proxy for measuring

collision risk. Perhaps this is no surprise. Reducing the maximum density in a vehicle's track is equivalent to expanding the sea room available to her. In extremity, more space to maneuver is always better.

5.2 Future Works

Possible future directions for this research are suggested below:

5.2.1 Continuous Density Analysis

Algorithm 2 in Chapter 2 can be extended to become a generalized look ahead algorithm that determines future contact density for all speed/heading values in the decision domain. This need not result in an explicit maneuvering recommendation, but could also produce alerts for current or planned maneuvers that result in maximum contact densities that are deemed high risk.

This type of alert system can help prevent future collisions. In the official Navy report on the collision between USS FITZGERALD (DDG 62) and motor vessel ACX CRYSTAL, it is pointed out that FITZGERALD "Were unaware of existing traffic separation schemes and the expected flow of traffic,...FITZGERALD's approved navigation track did not account for, nor follow, the Vessel Traffic Separation Schemes in the area" (TSS shown in Figure 5-1), [37]. Had such a system been onboard, it would have recognized and produced a notification that FITZGERALD's planned speed was leading her into a risky multi-ship close encounter.

5.2.2 Broadened Risk Analysis

This research uses contact density as a proxy measure for the risk of collision due to the possibility of speedy calculation. Future work could apply a more sophisticated risk model that evaluates maneuvering choices considering other factors such as contact geometry, type, or even historical behavior. An improved risk model can help the vessel to arrive at a high traffic density area with the minimal risk.

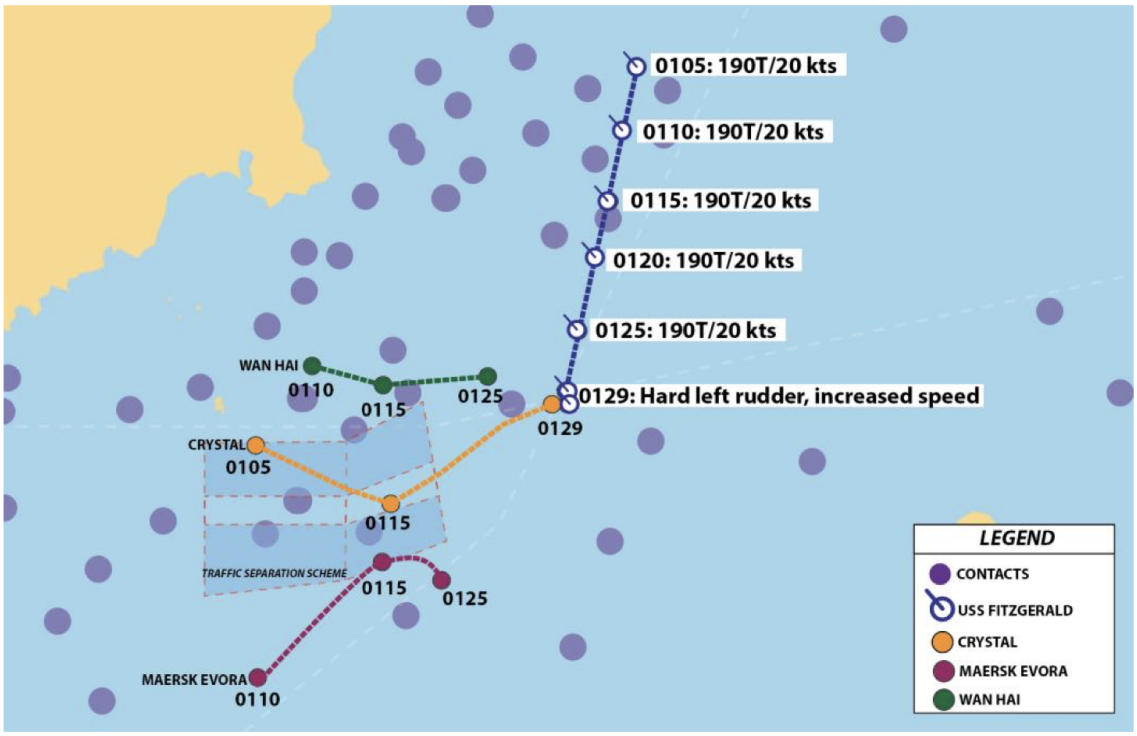


Figure 5-1: Illustration of the navigational situation of USS FITZGERALD collision with motor vessel ACX CRYSTAL. Figure from [37].

5.2.3 MOOS-IvP Specific

For MOOS developers interested in this research, the source code for the applications as well as mission files are available at the following GitHub repository:

```
https://github.com/jayzli/moos-ivp-jzli.git
```

Potential future developments could use auto-recognition of high density areas to activate the DensityCount behavior, incorporate multi-leg simulation into the pTrafficDensity applications, or further test these applications in different environments.

Appendix A

Acronyms and Terms used

AIS Automatic Identification System. A shipboard broadcast system that acts like a transponder.

CMVS Cooperative Multi-Vessel System. A system of vessels that uses communications to negotiate and collaborate with each other for the aim of improving overall safety, efficiency, or for performing specific tasks.

COLREGS The International Rules formalized in the Convention on the International Regulations for Preventing Collisions at Sea, 1972.

CPA Closest Point of Approach. An estimated point in which the distance between two objects, of which at least one is in motion, will reach its minimum value.

CRI Collision Risk Index. General term that applies to the numerical value of collision risk from a given model.

DCPA Distance to Closest Point of Approach. Also see CPA.

DWA Dynamic Window Algorithm. A collision avoidance algorithm that limits search space by considering the constraints of the robot's dynamics.

FDC Future Density Calculation. A algorithm proposed in this research that simulates various vehicle speeds and determines future maximum traffic

density.

FLTK Fast Light Toolkit. A cross-platform graphical control element library for graphical user interfaces.

IMO International Maritime Organization. A United Nations specialized agency with responsibility for the safety and security of shipping and the prevention of marine and atmospheric pollution by ships.

INS Integrated Navigation System. Modern navigation system installed on a ship's bridge that fuses sensor data overlays over electronic charts.

IPFPlot A MOOS-IvP built-in tool to view objective functions.

IvP Interval Programming. A popular multi-objective optimization technique.

MOOS Mission Oriented Operating Suite. A robotic middleware based on a publish-and subscribe architecture.

MOOSDB MOOS Data Base. The single database through which a MOOS community communicates .

OPAREA Operations Area. Or Operational Area. A bounded area where planned activities are taking place.

OpenGL A cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics.

PID ProportionalâĂŞIntegralâĂŞDerivative. Usually referring to the widely used feedback controller.

RRT Rapidly-exploring Random Trees. A popular algorithm for path finding.

SVM Support Vector Machines. A supervised learning model often used for classification and regression analysis.

TCPA Time to Closest Point of Approach. Also see CPA.

TSS Traffic Separation Scheme. A maritime route management system that divides heavy traffic waterways into lanes akin to highways.

VO Velocity Obstacle. The set of all velocities of a robot that will result in a collision with another robot.

USV Unmanned Surface Vehicle. One of many designations for surface marine robots. Also referred to as an Autonomous Surface Vehicle (ASV), Autonomous Marine Vehicle (AMV), etc.

ZAIC tool A MOOS-IvP built-in tool used to build one dimensional objective functions.

Appendix B

Detail Results of Selected Runs

Table B.1: Batch 5 Raw Results, without DensityCount

Run #	Encounters (70 m)	Average Encounter CPA (in m)	Near Misses (15 m)	Average Near Miss (in m)	Collisions (5 m)	Average Collision (in m)
1	295	45.6	0	0	0	0
2	237	41.93	0	0	0	0
3	253	42.19	1	12.83	0	0
4	202	47.18	0	0	0	0
5	213	44.77	0	0	0	0
6	178	42.98	0	0	0	0
7	238	44.98	2	10.65	0	0
8	221	39.68	0	0	0	0
9	219	45.03	0	0	0	0
10	194	45.7	0	0	0	0
11	295	40.99	0	0	0	0
12	186	43.12	0	0	0	0
13	225	42.81	2	13.43	0	0
14	186	46.43	0	0	0	0
15	253	40.63	4	12.91	0	0
16	199	45.67	0	0	0	0
17	254	43.84	0	0	0	0
18	178	41.2	0	0	0	0
19	208	46.12	0	0	0	0
20	103	38.5	0	0	0	0
21	284	45.31	0	0	0	0
22	222	43.16	0	0	0	0
23	144	43.03	0	0	0	0
24	273	45.34	0	0	0	0
25	157	43.08	0	0	0	0
26	236	45.62	0	0	0	0
27	169	44.89	0	0	0	0
28	261	43.05	2	14.68	0	0
29	138	42.33	0	0	0	0
30	220	42.27	0	0	0	0
31	180	38.83	0	0	0	0
32	254	41.42	1	10.83	0	0
33	140	45.67	0	0	0	0
34	256	43.03	0	0	0	0
35	133	42.28	0	0	0	0
36	216	42.88	0	0	0	0
37	121	43.47	0	0	0	0
38	248	46.71	0	0	0	0
39	190	43.93	0	0	0	0
40	218	48.95	0	0	0	0
41	165	45.14	0	0	0	0
42	174	45.05	0	0	0	0

Table B.2: Batch 5 Raw Results, with DensityCount and $c = 70$ m

Run #	Encounters (70 m)	Average Encounter CPA (in m)	Near Misses (15 m)	Average Near Miss (in m)	Collisions (5 m)	Average Collision (in m)
1	184	47.75	0	0	0	0
2	125	49.89	0	0	0	0
3	163	43.72	1	8.66	0	0
4	146	49.48	0	0	0	0
5	212	44.99	1	8.38	0	0
6	128	47.84	0	0	0	0
7	217	44.33	1	10.97	0	0
8	127	48.28	0	0	0	0
9	170	47.81	0	0	0	0
10	126	52.61	0	0	0	0
11	208	44.92	0	0	0	0
12	115	53.86	0	0	0	0
13	178	47.75	0	0	0	0
14	143	49.27	0	0	0	0
15	182	45.65	0	0	0	0
16	136	48.97	0	0	0	0
17	139	49.68	0	0	0	0
18	121	50.41	0	0	0	0
19	181	46.88	0	0	0	0
20	147	49.33	0	0	0	0
21	168	47.77	0	0	0	0
22	191	44.81	0	0	0	0
23	161	48.95	0	0	0	0
24	147	50.53	0	0	0	0
25	181	47.08	0	0	0	0
26	158	47.51	0	0	0	0
27	222	46.57	0	0	0	0
28	153	49.47	0	0	0	0
29	182	48.98	0	0	0	0
30	158	47.96	0	0	0	0
31	236	44.12	1	7.4	0	0
32	154	49.47	0	0	0	0
33	287	45.72	0	0	0	0
34	202	49.67	0	0	0	0
35	207	43.16	2	13.83	0	0
36	170	45.66	0	0	0	0
37	230	45.6	0	0	0	0
38	187	47.57	2	14.9	0	0
39	205	49.22	0	0	0	0
40	131	54.5	0	0	0	0
41	143	49.11	0	0	0	0
42	153	52.09	0	0	0	0

Table B.3: Batch 6 Raw Results, without DensityCount

Run #	Encounters (70 m)	Average Encounter CPA (in m)	Near Misses (15 m)	Average Near Miss (in m)	Collisions (5 m)	Average Collision (in m)
1	192	43.09	0	0	0	0
2	165	44.67	0	0	0	0
3	140	45.14	0	0	0	0
4	114	41.57	1	14.58	0	0
5	203	41.56	0	0	0	0
6	133	40.77	0	0	0	0
7	237	42.31	0	0	0	0
8	131	42.8	0	0	0	0
9	162	47.01	0	0	0	0
10	118	44.27	0	0	0	0
11	184	39.92	0	0	0	0
12	207	42.4	1	14.84	0	0
13	173	44.14	0	0	0	0
14	158	44.46	0	0	0	0
15	188	44.35	0	0	0	0
16	128	39.77	0	0	0	0
17	176	47.44	0	0	0	0
18	109	40.78	0	0	0	0
19	177	38.41	0	0	0	0
20	154	44.65	0	0	0	0
21	192	44.39	1	14.83	0	0
22	69	40.54	2	10.38	0	0
23	204	44.02	0	0	0	0
24	121	43.9	0	0	0	0
25	241	44.06	0	0	0	0
26	65	43.26	0	0	0	0
27	192	44.72	0	0	0	0
28	81	42.37	0	0	0	0
29	205	45.3	0	0	0	0
30	70	48.79	0	0	0	0
31	189	39.56	0	0	0	0
32	93	38.07	0	0	0	0
33	172	44.82	1	13.94	0	0
34	95	46.21	0	0	0	0
35	128	42.73	0	0	0	0
36	182	43.62	0	0	0	0
37	124	46	1	11.61	0	0
38	148	45.1	0	0	0	0
39	234	42.45	0	0	0	0
40	201	48.01	1	12.93	0	0
41	224	43.11	1	13.57	0	0
42	162	43.74	0	0	0	0
43	219	42.35	0	0	0	0
44	187	42.64	0	0	0	0

Table B.4: Batch 6 Raw Results, with DensityCount and $c = 70$ m

Run #	Encounters (70 m)	Average Encounter CPA (in m)	Near Misses (15 m)	Average Near Miss (in m)	Collisions (5 m)	Average Collision (in m)
1	119	50.54	0	0	0	0
2	115	51.94	0	0	0	0
3	112	52.05	0	0	0	0
4	107	46.84	0	0	0	0
5	90	50.61	0	0	0	0
6	122	46.4	0	0	0	0
7	86	50.8	0	0	0	0
8	122	46.94	0	0	0	0
9	104	48.56	0	0	0	0
10	98	46.78	0	0	0	0
11	94	54.4	0	0	0	0
12	121	48.2	0	0	0	0
13	97	51.41	0	0	0	0
14	113	45.09	0	0	0	0
15	97	51.59	0	0	0	0
16	98	50.76	0	0	0	0
17	75	56.51	0	0	0	0
18	120	51.14	0	0	0	0
19	64	49.57	0	0	0	0
20	120	48.24	0	0	0	0
21	130	46.87	0	0	0	0
22	169	48.24	0	0	0	0
23	101	50.42	0	0	0	0
24	128	48.14	0	0	0	0
25	125	48.67	0	0	0	0
26	196	45.57	0	0	0	0
27	178	47.26	0	0	0	0
28	171	50.87	0	0	0	0
29	160	50.25	0	0	0	0
30	133	48.11	0	0	0	0
31	158	50.06	0	0	0	0
32	134	48.95	0	0	0	0
33	101	47.61	0	0	0	0
34	82	49.95	0	0	0	0
35	135	46.09	0	0	0	0
36	115	52.2	0	0	0	0
37	159	46.63	0	0	0	0
38	139	51.03	0	0	0	0
39	102	51.96	0	0	0	0
40	127	47.21	0	0	0	0
41	95	50.32	0	0	0	0
42	136	44.32	0	0	0	0
43	96	50.67	0	0	0	0
44	130	50.42	0	0	0	0
45	122	48.91	0	0	0	0
46	113	45.81	0	0	0	0

Bibliography

[1] Jin Hyeong Ahn, Key Pyo Rhee, and Young Jun You. A study on the collision avoidance of a ship using neural networks and fuzzy logic. *Applied Ocean Research*, 37:162–173, 2012.

[2] Craig H. Allen. *Farewell's Rules of the Nautical Road*. Naval Institute Press, Annapolis, Maryland, eighth edition, 2005.

[3] Michael Benjamin. Marine autonomy, sensing and communication, Lecture one.

[4] Michael Benjamin. References in the recent literature using MOOS-IvP (May 2019).

[5] Michael R. Benjamin. MIT-CSAIL-TR-2004-058. Technical report, Massachusetts Institute of Technology, 2004.

[6] Michael R. Benjamin. MOOS-IvP Autonomy Tools Users Manual Release 13.2. Technical report, MIT Computer Science and Artificial Intelligence Lab, www.moos-ivp.org/docs/, February 2013.

[7] Michael R. Benjamin. Nested Autonomy. *Journal of Field Robotics*, pages 1–17, 2014.

[8] Michael R Benjamin. Computer Science and Artificial Intelligence Laboratory Technical Report Autonomous COLREGS Modes and Velocity Functions Autonomous COLREGS Modes and Velocity Functions. Technical report, Massachusetts Institute of Technology, 2017.

[9] Michael R. Benjamin. Fast-CPA: A layered caching algorithm for rapid closest point of approach calculations in marine collision avoidance. In *OCEANS 2017 MTS/IEEE Anchorage*, Anchorage, AK, September 2017.

[10] Michael R. Benjamin. Capturing velocity function plateaus for efficient marine vehicle collision avoidance calculations. *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans, OCEANS - Kobe 2018*, 2018.

[11] Michael R. Benjamin, John J. Leonard, Joseph A. Curcio, and Paul M. Newman. A Method for Protocol-Based Collision Avoidance between Autonomous Marine Surface Craft. *Journal of Field Robotics*, 23(5):1–15, 2006.

[12] Krishnadev Calamur. High traffic, high risk in the strait of malacca.

[13] United States Coast Guard Navigation Center. *Navigation rules, international-inland*. Department of Homeland Security, United States Coast Guard, Washington, DC, August 2014.

[14] Linying Chen, Hans Hopman, and Rudy R. Negenborn. Distributed model predictive control for vessel train formations of cooperative multi-vessel systems. *Transportation Research Part C: Emerging Technologies*, 92(May):101–118, 2018.

[15] Pengfei Chen, Yamin Huang, Junmin Mou, and P. H.A.J.M. van Gelder. Probabilistic risk analysis for ship-ship collision: State-of-the-art. *Safety Science*, 117(September 2018):108–122, 2019.

[16] Zheng Chen, Youming Zhang, Yougong Zhang, Yong Nie, Jianzhong Tang, and Shiqiang Zhu. A Hybrid Path Planning Algorithm for Unmanned Surface Vehicles in Complex Environment with Dynamic Obstacles. *IEEE Access*, 7:126439–126449, 2019.

[17] Robert Girrier and James Stavridis. *Watch officers guide: a handbook for all deck watch officers*. Naval Inst. Press, 2007.

[18] Floris Goerlandt and Pentti Kujala. On the reliability and validity of ship-ship collision risk analysis in light of different perspectives on risk. *Safety Science*, 62:348–365, 2014.

[19] Floris Goerlandt, Jakub Montewka, Vladimir Kuzmin, and Pentti Kujala. A risk-informed ship collision alert system: Framework and application. *Safety Science*, 77:182–204, 2015.

[20] Kiyoshi Hara and Shinya Nakamura. A comprehensive assessment system for the maritime traffic environment. *Safety Science*, 19(2-3):203–215, 1995.

[21] Helmut Hilgert and Michael Baldauf. A common risk model for the assessment of encounter situations on board ships. *Deutsche Hydrographische Zeitschrift*, 1997.

[22] Yamin Huang, Linying Chen, Pengfei Chen, Rudy R. Negenborn, and P. H.A.J.M. van Gelder. Ship collision avoidance methods: State-of-the-art. *Safety Science*, 121(August 2019):451–473, 2020.

[23] Yamin Huang, Linying Chen, and P. H.A.J.M. van Gelder. Generalized velocity obstacle algorithm for preventing ship collisions at sea. *Ocean Engineering*, 173(November 2018):142–156, 2019.

[24] Yamin Huang, Linying Chen, and P. H.A.J.M. van Gelder. Generalized velocity obstacle algorithm for preventing ship collisions at sea. *Ocean Engineering*, 173(November 2018):142–156, 2019.

[25] Tor Arne Johansen, Tristan Perez, and Andrea Cristofaro. Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment. *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[26] Sheng Long Kao, Kuo Tien Lee, Ki Yin Chang, and Min Der Ko. A fuzzy logic method for collision avoidance in vessel traffic service. *Journal of Navigation*, 2007.

[27] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings - IEEE International Conference on Robotics and Automation*, 1985.

[28] Katsuro Kijima and Yoshitaka Furukawa. Automatic collision avoidance system using the concept of blocking area. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 36(21):223–228, 2003.

[29] Przemysław Krata and Jakub Montewka. Assessment of a critical area for a give-way ship in a collision encounter. *Archives of Transport*, 34(2):51–60, 2015.

[30] D. K.M. Kufoalor, T. A. Johansen, E. F. Brekke, A. Hepsø, and K. Trnka. Autonomous maritime collision avoidance: Field verification of autonomous surface vehicle behavior in challenging scenarios. *Journal of Field Robotics*, 37(3):387–403, 2020.

[31] Yoshiaki Kuwata, Michael T. Wolf, Dimitri Zarzhitsky, and Terrance L. Huntsberger. Safe maritime navigation with COLREGS using velocity obstacles. *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4728–4734, 2011.

[32] Xinyu Liu, Yun Li, Jing Zhang, Jian Zheng, and Chunxi Yang. Self-Adaptive Dynamic Obstacle Avoidance and Path Planning for USV Under Complex Maritime Environment. *IEEE Access*, 7:114945–114954, 2019.

[33] Yu Hong Liu and Hong Xia Liu. Case learning base on evaluation system for vessel collision avoidance. In *Proceedings of the 2006 International Conference on Machine Learning and Cybernetics*, 2006.

[34] Adan Lopez-Santander and Jonathan Lawry. An Ordinal Model of Risk Based on Mariner's Judgement. *Journal of Navigation*, 70(2):309–324, 2017.

[35] Hongguang Lyu and Yong Yin. COLREGS-Constrained Real-Time Path Planning for Autonomous Ships Using Modified Artificial Potential Fields. *Journal of Navigation*, 2019.

[36] Jun Min Mou, Cees van der Tak, and Han Ligteringen. Study on collision avoidance in busy waterways by using AIS data. *Ocean Engineering*, 37(5-6):483–490, 2010.

[37] Office of the Chief of Naval Operations. Report on the collision between USS FITZGERALD (DDG 62) and motor vessel ACX CRYSTAL. Memorandum for Distribution, October 2017.

[38] Ulku Ozturk and Kadir Cicek. Individual collision risk assessment in ship navigation: A systematic literature review. *Ocean Engineering*, 180(April):130–143, 2019.

[39] L. P. Perera, J. P. Carvalho, and C. Guedes Soares. *Bayesian network based sequential collision avoidance action execution for an ocean navigational system*, volume 43. IFAC, 2010.

[40] Lokukaluge P. Perera and C. Guedes Soares. Collision risk detection and quantification in ship navigation with integrated bridge systems. *Ocean Engineering*, 109:344–354, 2015.

[41] R. Śmierzchalski. Ships' domains as a collision risk at sea in the evolutionary trajectory planning. In *Management Information Systems*, 2000.

[42] Rui Song, Yuanchang Liu, and Richard Bucknall. Smoothed A* algorithm for practical unmanned surface vehicle path planning. *Applied Ocean Research*, 2019.

[43] James Stavridis. *Sea Power - The History and Geopolitics of the World's Oceans*. Penguin Books, New York, New York, first edition, 2017.

[44] Martin Svanberg, Vendela Santén, Axel Hörteborn, Henrik Holm, and Christian Finnsgård. AIS in maritime research. *Marine Policy*, 106(December 2018):103520, 2019.

[45] Rafal Szlapczynski, Przemyslaw Krata, and Joanna Szlapczynska. Ship domain applied to determining distances for collision avoidance manoeuvres in give-way situations. *Ocean Engineering*, 165(March):43–54, 2018.

[46] Rafal Szlapczynski and Joanna Szlapczynska. A method of determining and visualizing safe motion parameters of a ship navigating in restricted waters. *Ocean Engineering*, 129(November 2016):363–373, 2017.

[47] Pingpeng Tang, Rubo Zhang, Deli Liu, Lihua Huang, Guanqun Liu, and Tingquan Deng. Local reactive obstacle avoidance approach for high-speed unmanned surface vehicle. *Ocean Engineering*, 2015.

[48] the International Register of Shipping (INTLREG). IMO navigation rules at Straits of Malacca and Singapore, Dec 2019.

[49] K. L. Woerner. Colregs-compliant autonomous collision avoidance using multi-objective optimization with interval programming. Ma thesis, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139:Massachusetts Institute of Technology, June 2014.

[50] Kyle Woerner, Michael R. Benjamin, Michael Novitzky, and John J. Leonard. Quantifying protocol evaluation for autonomous collision avoidance: Toward establishing COLREGS compliance metrics. *Autonomous Robots*, 43(4):967–991, 2019.

[51] Jinfen Zhang, Di Zhang, Xinping Yan, Stein Haugen, and C. Guedes Soares. A distributed anti-collision decision support formulation in multi-ship encounter situations under COLREGs. *Ocean Engineering*, 105:336–348, 2015.

[52] Weibin Zhang, Floris Goerlandt, Pentti Kujala, and Yinhai Wang. An advanced method for detecting possible near miss ship collisions from AIS data. *Ocean Engineering*, 124:141–156, 2016.